NASA CONTRACTOR
REPORT

NASA CR-161328

STANDARD TRANSISTOR ARRAY (STAR) - SIMLOG TESTGN PROGRAMMER'S GUIDE, Volume 2, Addendum 2

By B. D. Carroll
Electrical Engineering Department
Auburn University
Auburn, Alabama 36830

FOR EARLY DOMESTIC DISSEMINATION

Date for general release: November 1981

Final Report

September 14, 1979

Prepared for

NASA - GEORGE C. MARSHALL SPACE FLIGHT CENTER
Marshall Space Flight Center, Alabama 35812

TECHNICAL REPORT STANDARD TITLE PAGE

1 REPORT NO	2 GOVERNMENT ACCESSION NO.	3 RECIPIENT'S CATALOG NO.
NASA CR-161328		

4 TITLE AND SUBTITLE	5 REPORT DATE
Standard Transistor Array (STAR) - SIMLOG/TESTGN Programmer's Guide. Final Report, Volume 2, Addendum 2	September 14, 1979
	6 PERFORMING ORGANIZATION CODE
7 AUTHOR(S) B. D Carroll	8 PERFORMING ORGANIZATION REPORT #
9 PERFORMING ORGANIZATION NAME AND ADDRESS Electrical Engineering Department Auburn University Auburn, Alabama 36830	10 WORK UNIT NO.
	11 CONTRACT OR GRANT NO. NAS8-31572
12 SPONSORING AGENCY NAME AND ADDRESS National Aeronautics and Space Administration Washington, DC 20546	13 TYPE OF REPORT & PERIOD COVERED Contractor Report Final
	14 SPONSORING AGENCY CODE

15 SUPPLEMENTARY NOTES

This work was done under the technical supervision of Mr. Jack Matheney, George C. Marshall Space Flight Center, Alabama.

16. ABSTRACT

A brief introduction to the SIMLOG/TESTGN system of programs is given. SIMLOG is a logic simulation program, whereas TESTGN is a program for generating test sequences from output produced by SIMLOG.

The structures of the two programs are described. Data base, main program, and subprogram details are also given. Guidelines for program modifications are discussed. Commented program listings are included.

17 KEY WORDS	18. DISTRIBUTION STATEMENT
	FOR EARLY DOMESTIC DISSEMINATION J. Brooks Moore 11/20/79 F. BROOKS MOORE Dir, Electronics and Control Laboratory, MSFC

19 SECURITY CLASSIF (of this report)	20 SECURITY CLASSIF. (of this page)	21 NO. OF PAGES	22 PRICE
Unclassified	Unclassified	136	NTIS

MSFC - Form 3292 (Rev December 1972)

For sale by National Technical Information Service, Springfield, Virginia 22151

Page intentionally left blank

Page intentionally left blank

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The purpose of this document is to provide programming level detail of the SIMLOG/TESTGN system. Those interested in the theoretical basis of the system should refer to [1]. User information on the system is given in [2].

A brief functional description of the SIMLOG/TESTGN system is presented below. Details of the system structure are given in Section 2. The system database is described in Section 3 while the program descriptions are given in Section 4. Modification guidelines can be found in Section 5. Heavily commented listings of all programs are included as an Appendix.

The system of programs described herein was written to provide a cost-effective but powerful tool for simulating logic circuits and for generating test sequences for logic circuits. BASIC was chosen as the implementation language since it is commonly available on minicomputer-based and microcomputer-based computing systems. Application of the programs to circuits containing five-hundred or more gates is feasible if the host computer system provides a sufficient primary and secondary storage capacity.

SIMLOG is a gate-level logic simulation program that is based on a three-valued logic model and on a unit-delay timing model. Both combinational and sequential circuits can be handled. Circuits containing

single or multiple stuck-type faults can be simulated. Logic elements available for use include NAND gates, NOR gates, unit delay elements, and edge-triggered D flip-flops.

TESTGN is a test sequence generation program for circuits previously simulated by SIMLOG. The program can be used to generate tests for unspecified faults or for specified single stuck-type faults.

2. SYSTEM STRUCTURE

The top-level structure of the SIMLOG/TESTGN system is shown in Figure 1. Both programs build portions of the database but that segment built by SIMLOG must be available prior to the execution of TESTGN. The SIMLOG database is self-generated.

Two versions of the system have been written. One version is written in Xerox BASIC for execution on a Sigma 5 CPV computer system. The second version is written in BASIC-PLUS for execution on a PDP 11/40 RSTS/E computer system.

The structure of SIMLOG is different for the two versions, while the TESTGN structure is common to both versions. Figure 2(a) shows the SIMLOG structure for the PDP 11 version. The structure of the Sigma 5 version is detailed in Figure 2(b). Differences in the two structures are necessary due to array access methods of the two host computers. Additional details on these differences are given in Section 3. Correspondence between the various subprograms is as follows. SIMA and SIM1 are equivalent, whereas SIMB combines the functions of SIM2 and RACE.

Each SIMLOG module or subprogram is written as a separate program. Chaining is used to transfer control and execution from one subprogram to another. TESTGN is implemented as a single program module. A lower level structural description of each program module will now be given.

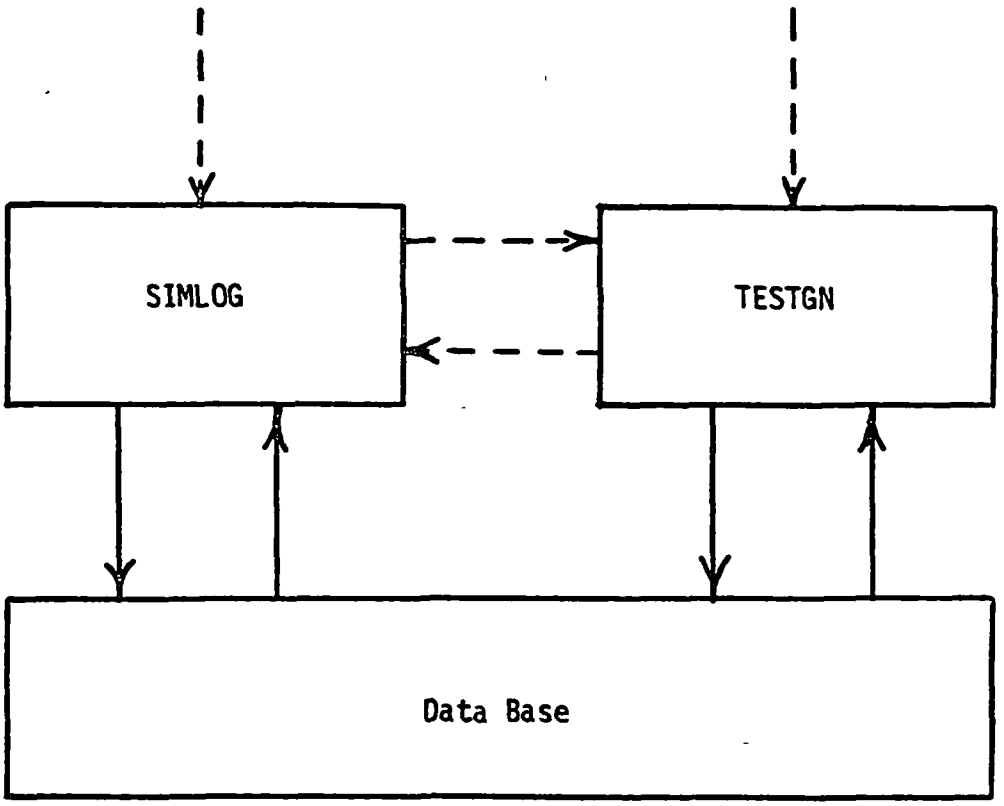
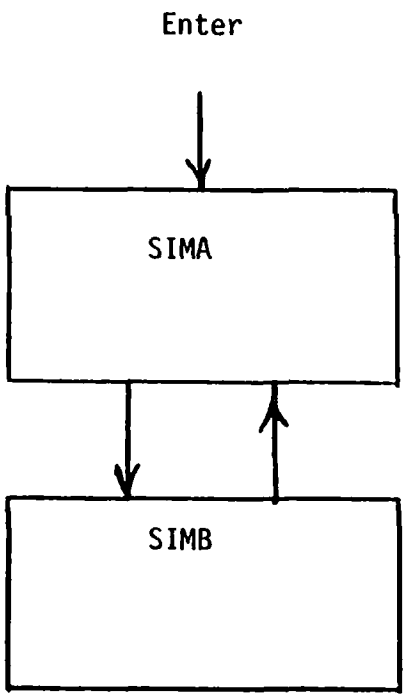
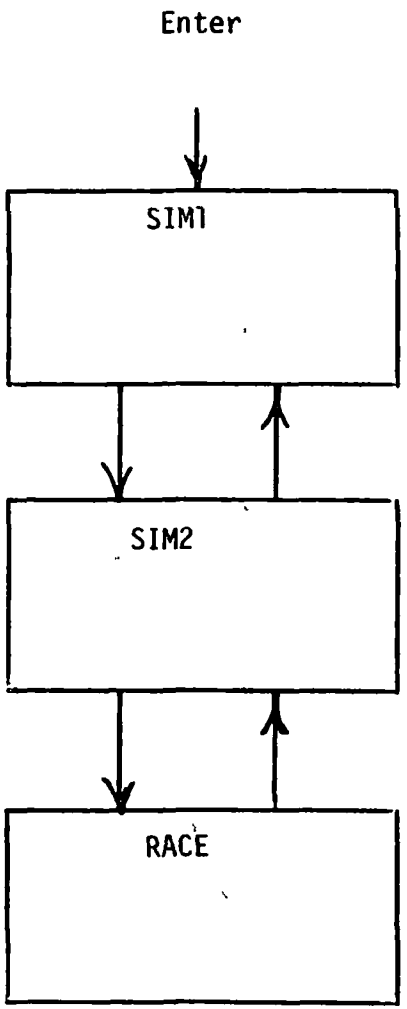


Figure 1 - SIMLOG/TESTGN System Structure.



(a)

SIMLOG - PDP 11/40



(b)

SIMLOG - SIGMA5

Figure 2. SIMLOG STRUCTURE

Figure 3 shows the structure of the SIMA and SIM1 subprograms. The SIMB structure is given in Figure 4; while the structures of SIM2 and RACE are given in Figures 5 and 6, respectively. See Figure 7 for the structure of TESTGN.

It is clear that the subprograms share a number of common routines. All routines are described in detail in Section 4. The next section is devoted to a description of the system database.

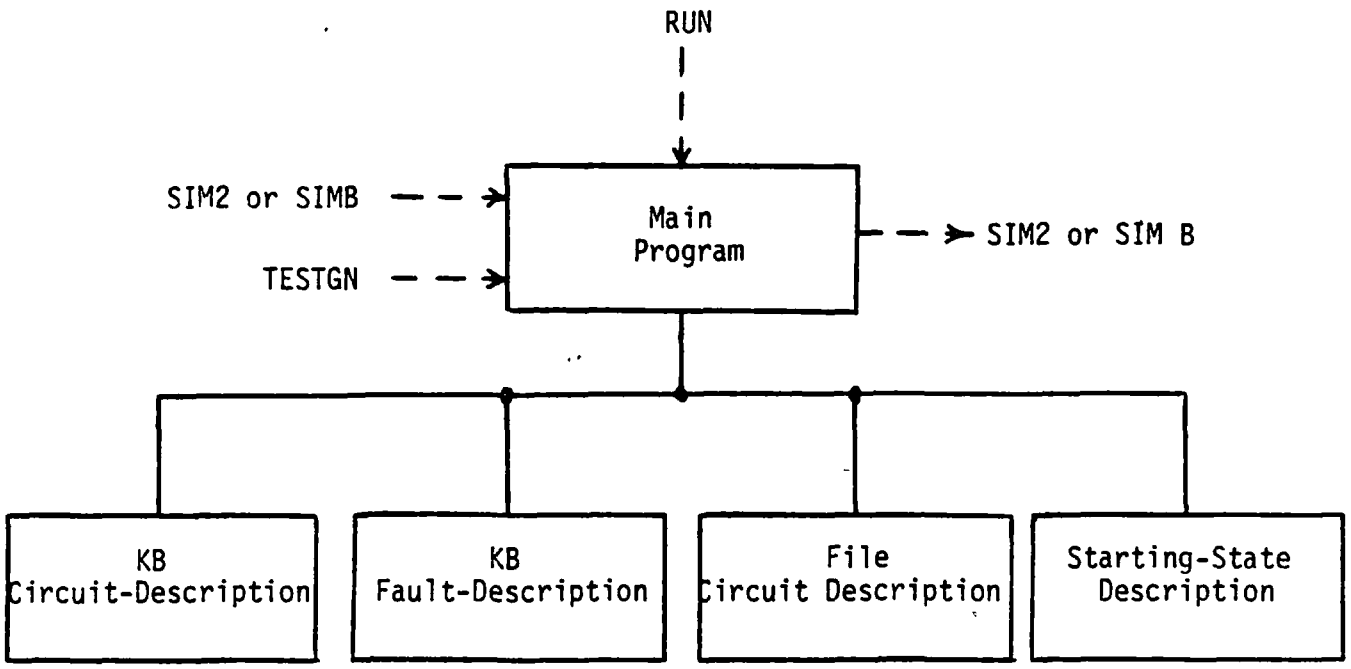


Figure 3. SIM1 and SIMA Structure

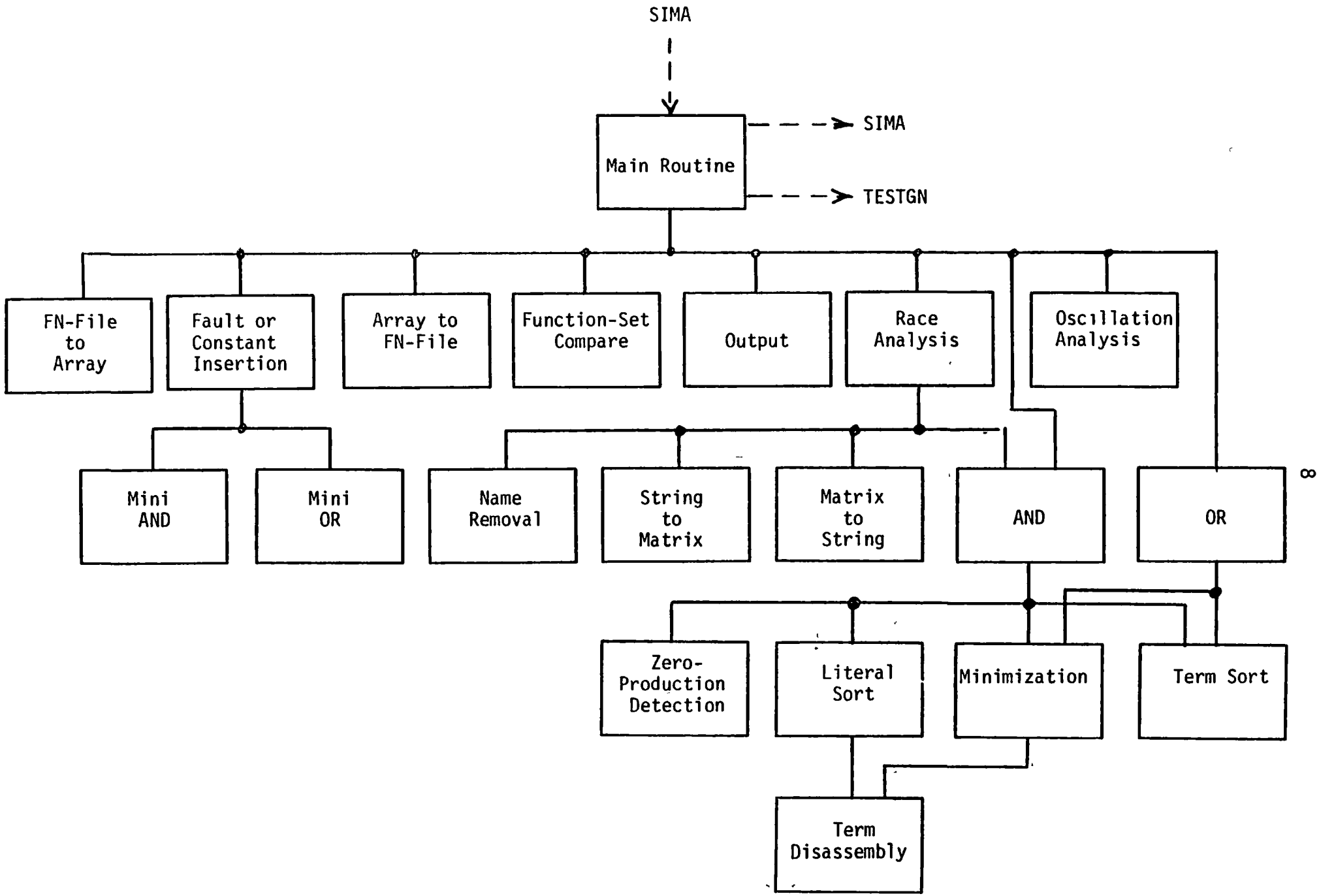


Figure 4. SIMB Structure

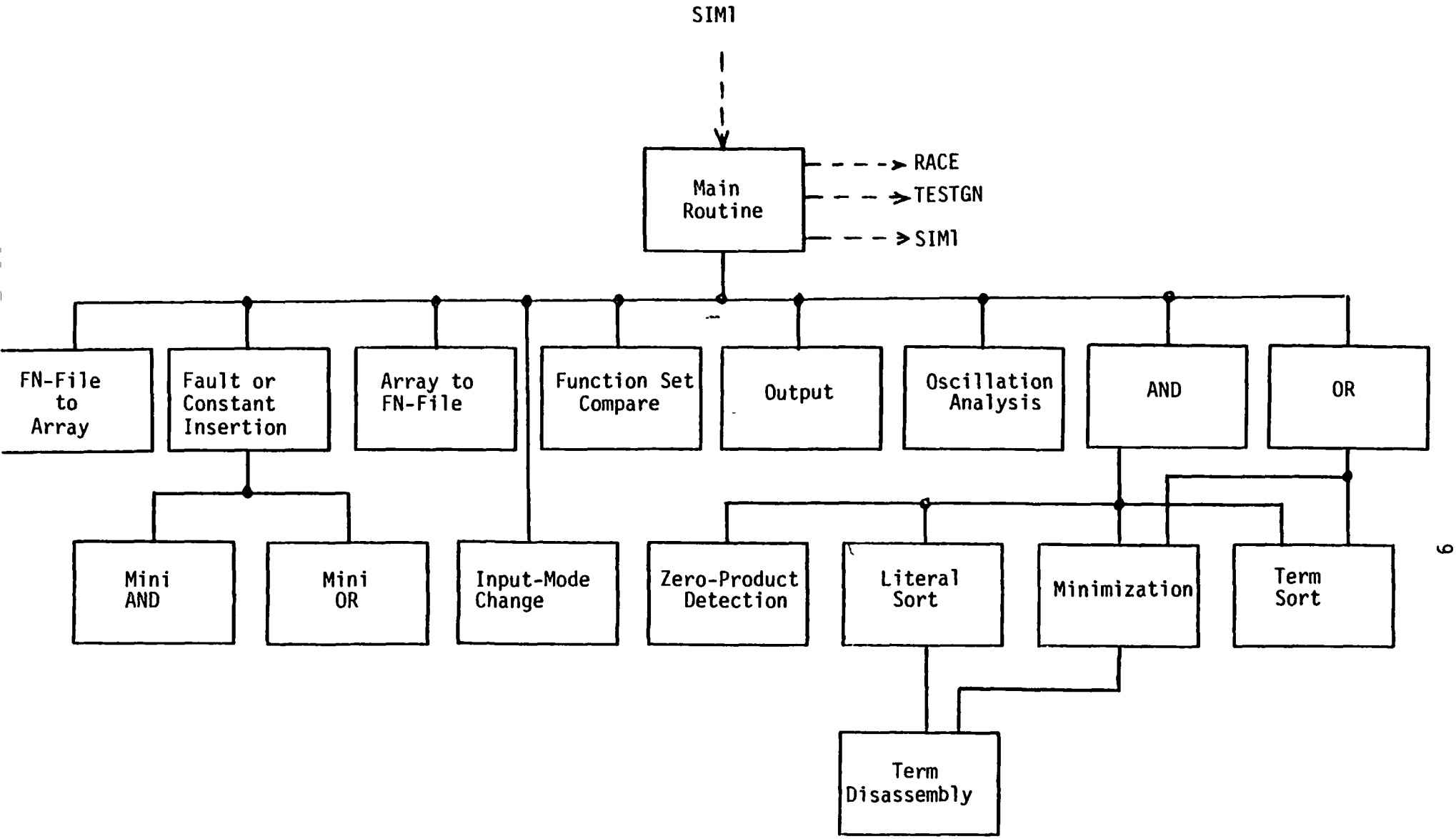


Figure 5. SIM2 Structure.

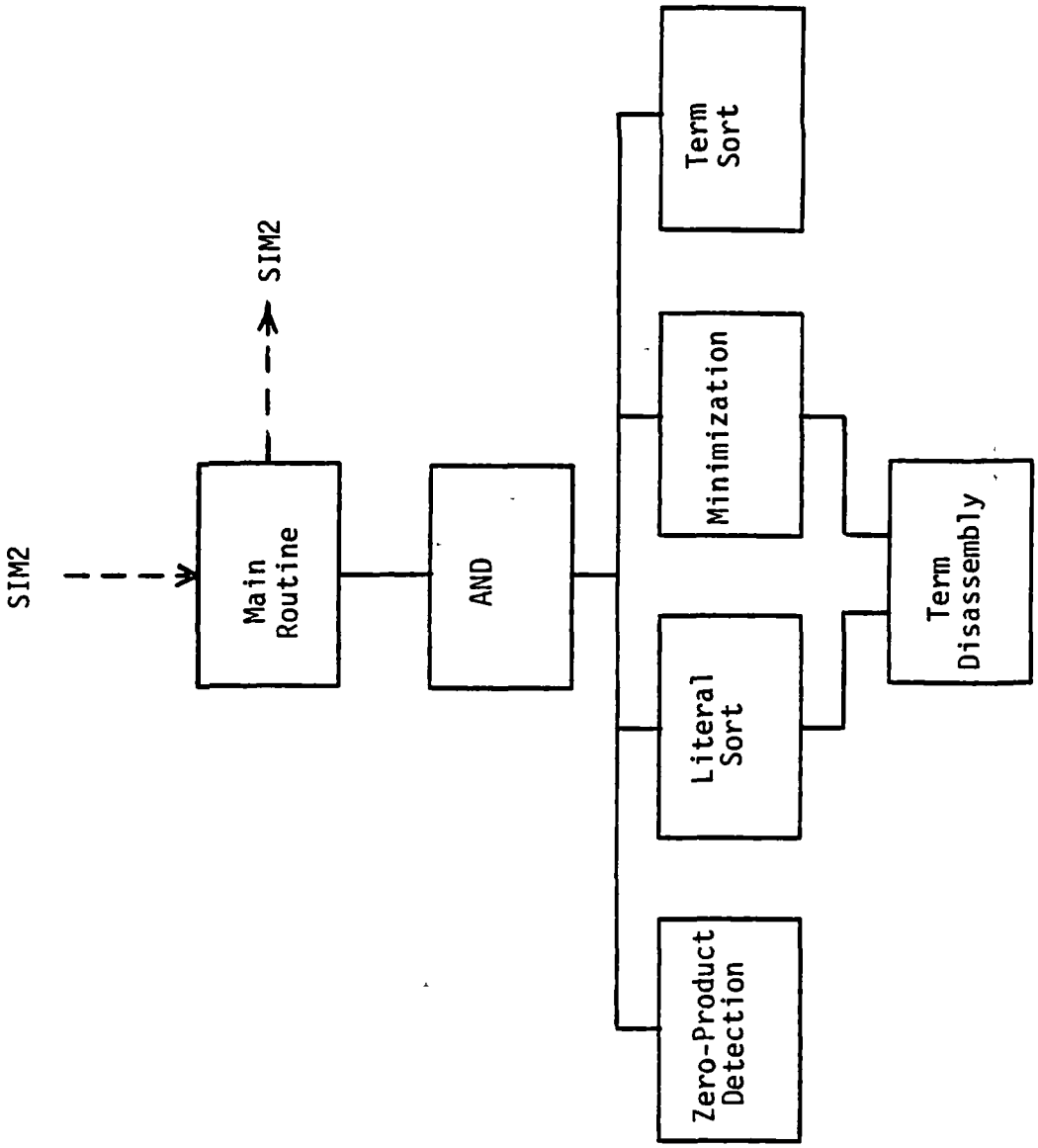


Figure 6. RACE Structure.

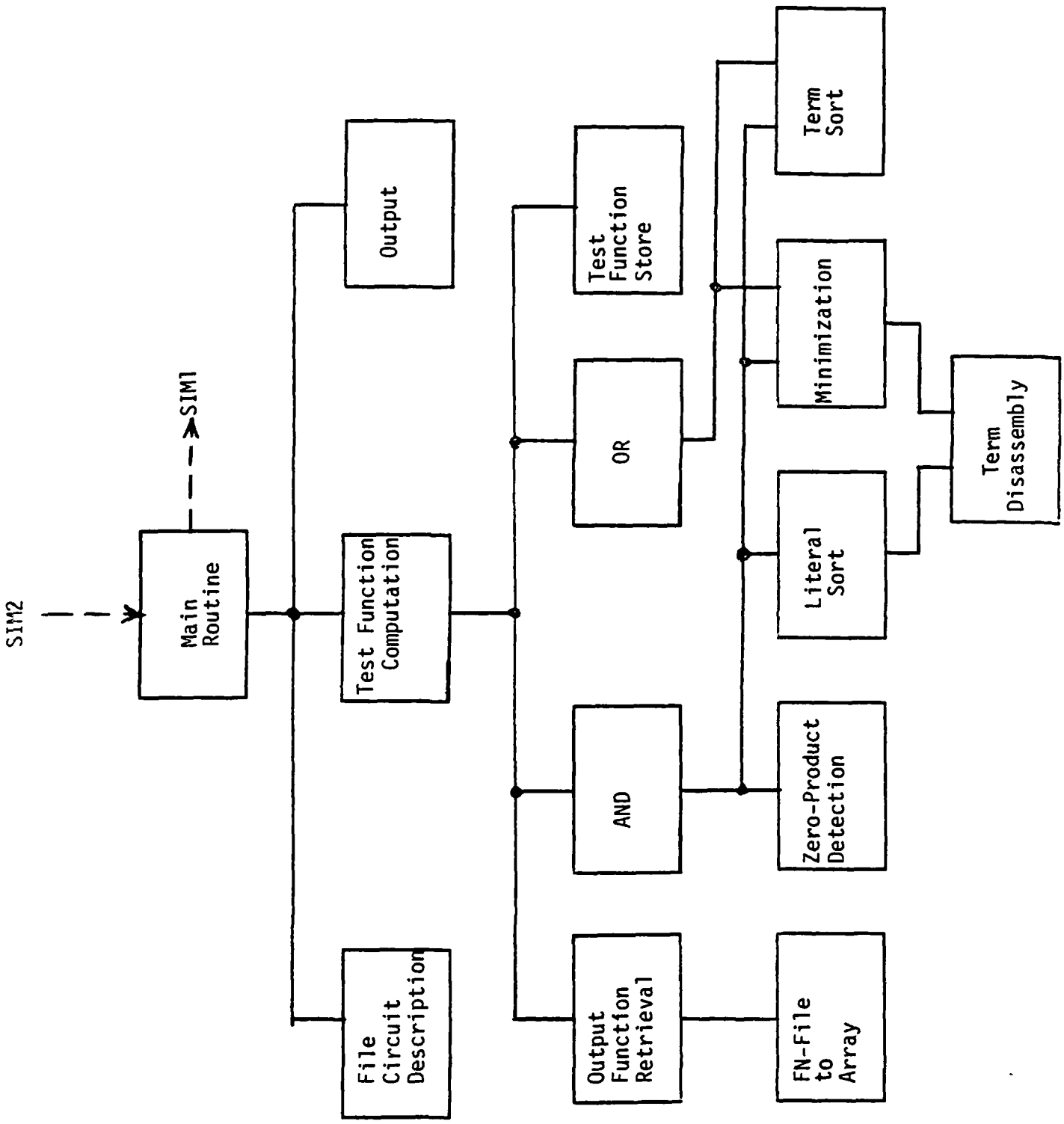


Figure 7. TESTGN Structure.

3. SYSTEM DATABASE

The SIMLOG/TESTGN database consists of structures for storing a description of the circuit to be processed, for storing the function set generated for a given ripple-time, and for storing operand and resultant functions. Each of these structures is described below. Other related matters such as I/O stream assignments are also discussed in this section. In the following descriptions, the array and file names used are common to both versions of SIMLOG unless indicated by an alternate name enclosed within parentheses. In such a case, the term in parentheses corresponds to the PDP 11 version, and the preceding term to the Sigma 5 version.

CIRCUIT DESCRIPTION ARRAYS

The description of a circuit to be processed by SIMLOG or TESTGN is contained in arrays N(N$), P, I(I$), and Q(A$) which are described below.

N(N$). This is an array of character strings with dimensions N x 4 where N is the number of elements in the circuit. Each row in the array corresponds to a particular element and the row number is the same as the record number in the data files where the gate output equations are stored. Figure 8(a) describes the general format of the array while Figure 8(b) illustrates the array for the circuit shown in Figure 9.

		1	2	3	4
	1	Gate	Gate	Fault	Circuit
Records	2 : N	Name	Type	Condition	Output or Not

(a) General Format

	1	2	3	4
1	A	NAND	FF	NO
2	B	NAND	FF	NO
3	C	NAND	FF	NO
4	D	NAND	FF	NO
5	E	NAND	FF	NO
6	F	NAND	FF	NO
7	G	NAND	FF	NO
8	Q	NAND	FF	YES
9	QBAR	NAND	FF	YES

(b) Example from Figure 9

Figure 8. Array N(N$) Format.

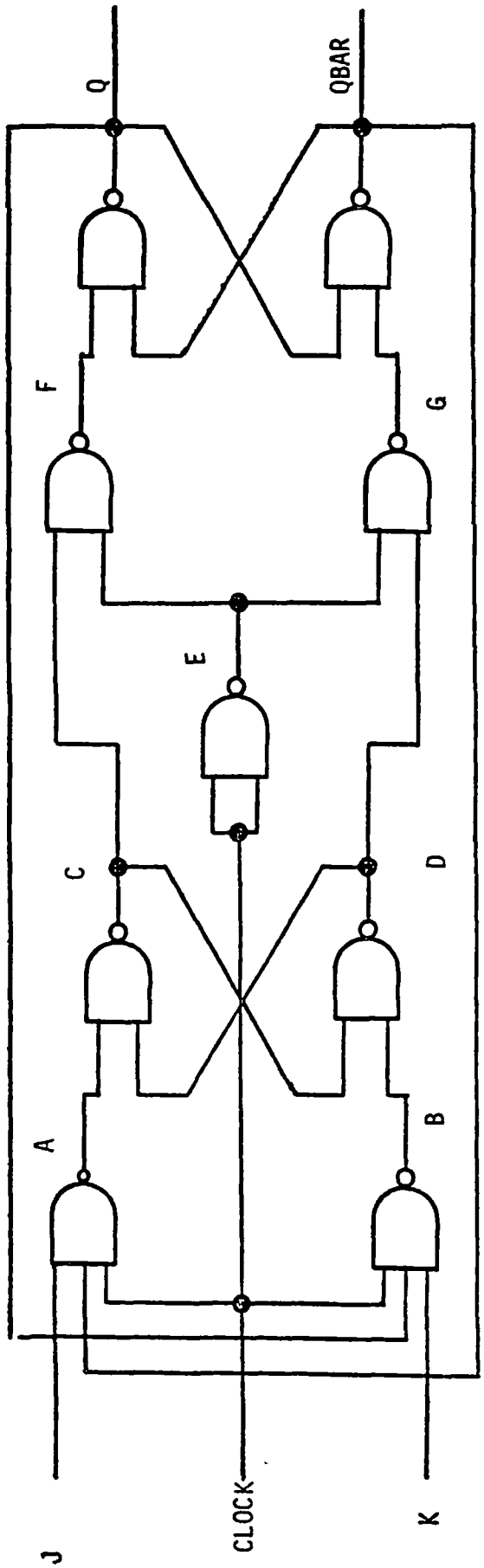


Figure 9. JK Flip-Flop Circuit

P. Array P is an N x 2 numerical array used in conjunction with array I(I$) for describing element inputs. The array format is described and illustrated in Figure 10.

I(I$). This is an M x 3 string array used in conjunction with array P to describe element input lines. Figure 11 shows the format of the array. Note that M is equal to the total of all element fan-in.

Q(Q$). This is an L x 2 string array that indicates pairs of cross-coupled gates. L is the number of cross-coupled gates in the circuit. See Figure 12 for the detailed description of this array.

FUNCTION FILES

Equations produced during a SIMLOG simulation run are represented as character strings. Each equation is stored as a separate and distinct record in a file. Four files are needed for storing all of the equations produced and are described in Table 1. Each element in a circuit is assigned a record number corresponding to its N(N$) array row number. Functions are stored in sum-of-products form with period used to indicate the end.

Function file access switching is used to effect the transfer of functions in the NEW0 and NEW1 files to the OLD0 and OLD1 files for the next ripple-time step. This switching is implemented by use of a switch S which is initialized to the value 0. When S=0, functions computed in the previous ripple-time (r-1) are accessed from OLD0 and OLD1. When S=1, functions computed in the previous ripple-time (r-1) are accessed from NEW0 and NEW1. The value of S is complemented at the end of each ripple-time step.

	1	2
1	Number of Gate Inputs (Fan-in)	Input List Pointer to Array I(I$)
2		
:		
N		

(a) General Format

	1	2
1	3	1
2	3	4
3	2	7
4	2	9
5	2	11
6	2	13
7	2	15
8	2	17
9	2	19

(b) Example from Figure 9

Figure 10. Description of Array P.

	1	2	3
1	Input Names	Fault Condition	Circuit Input Flag or Net Number
2			
:			
M			

(a) General Format

	1	2	3
1	J	FF	YES
2	CLOCK	FF	YES
3	QBAR	FF	9
4	K	FF	YES
5	CLOCK	FF	YES
6	Q	FF	8
7	A	FF	1
8	D	FF	4
9	B	FF	2
10	C	FF	3
11	CLOCK	FF	YES
12	C	FF	3
13	E	FF	5
14	D	FF	4
15	E	FF	5
16	F	FF	6
17	QBAR	FF	9
18	G	FF	7
19	Q	FF	8

(b) Example from Figure 9

Figure 11. Description of Array I(I$).

	1	2
1	First Gate	Second Gate
2	Output name	Output Name
⋮		
L		

(a) General Format

	1	2
1	C	D
2	Q	QBAR

(b) Example from Figure 9

Figure 12. Description of Array Q(Q$).

TABLE 1. Data File Definitions

File Name	File Contents
OLDØ	Contains records that describe the Ø-equations produced during the previous ripple-time step.
OLD1	Contains records that describe the 1-equations produced during the previous ripple-time step.
NEWØ	Contains records that describe the Ø-equations produced during the current ripple-time step.
NEW1	Contains records that describe the 1-equations produced during the current ripple-time step.

PROCESSING ARRAYS

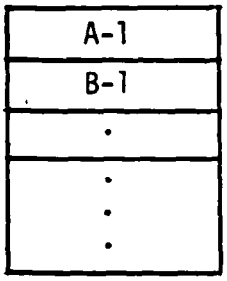
Functions to be processed are retrieved from the appropriate function file and are copied to a temporary array or its equivalent for input to the appropriate processing routine. Only the expression to the right of the equal sign is copied to the temporary structure. The output from a processing routine is placed in a like temporary structure prior to further processing or to storage in a function file. These temporary structures will now be described for each version of SIMLOG. A major difference in the two versions is found in this area.

PDP 11 Version

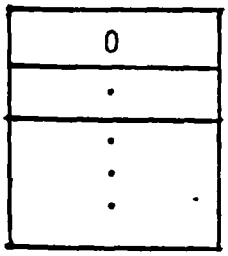
The temporary arrays discussed above are implemented using the virtual array capability of BASIC-PLUS. This approach yields the flexibility of standard arrays without requiring a large segment of primary memory space for the arrays. The arrays are described below.

F$ and G$. These are one dimension string arrays used to hold expressions that are the operands for a processing routine such as the AND routine. Each product term of an expression is assigned to a unique element of an array. The period is used to indicate end-of-function. Constant expressions (0 or 1) are represented by the corresponding symbol in element one of the array. Functions X1=A-1+B1, Y-1=0, and Y1=1 are represented as shown in Figures 13(a), (b), (c), respectively.

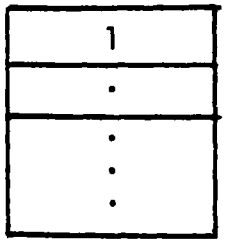
H$. A one dimensional string array used to store the output or resultant of a processing routine. H$ has the same structure as F$ and G$.



(a) X1 = A-1 + B1



(b) Y-1 = 0



(c) Y1 = 1

Figure 13. Function Representation in Arrays

Sigma 5 Version

Virtual arrays are not avialable with Xerox BASIC. Also, the implementation of the processing arrays as standard arrays is not feasible due to the large amount of primary memory required. Hence, the processing arrays are realized in the Sigma 5 version as files, herein referred to as processing-array files. The file names are FARRAY, GARRAY, and HARRAY.

Processing-array files are easily accessed by taking advantage of the record I/O mode of Xerox BASIC. For example, the equivalent of X$=F$(Y) can be performed by a statement of the following form.

```
INPUT  :F;Y,X$
```

where F represents the I/O stream to which FARRAY has been opened. Also, G$(Y)=X$ is accomplished with

```
PRINT  :G;Y,X$
```

where like before G represents the I/O stream to which GARRAY has been opened.

I/O STREAM (CHANNEL) ASSIGNMENTS

Files must be opened to an I/O stream (Sigma 5 terminology) or I/O channel (PDP 11 terminology) before they can be accessed. Assignment of channels in the PDP 11 version is static since at most six channels of the available twelve channels are needed. On the other hand, the assignment in the Sigma 5 version must be dynamic since only four streams are available while up to eight are needed. Table 2 shows

TABLE 2

SIMA and SIM1 Channel Assignments

Channel or Stream	File
1	OLD1
2	OLDO
3	NET

the assignments for subprograms SIMA and SIM1 in both versions. Table 3 shows the assignments for SIMB while Table 4 shows the assignments for SIM2 and RACE. Note that the latter assignment is a function of the file-switch S and of the function-switch J.

TABLE 3

SIMB I/O Channel Assignments (PDP 11)

Channel	File	
1	OLD1	Function Files
2	OLDØ	
3	NEWØ	
4	NEW1	
5	NET	Circuit Description
6	VIR	Virtual Arrays

TABLE 4

SIM2 and RACE I/O Stream Assignments (Sigma 5)

	S=0		S=1	
Stream	J=0	J=1	J=0	J=1
1	OLD1	FARRAY	FARRAY	GARRAY
2	FARRAY	OLDO	GARRAY	FARRAY
3	GARRAY	HARRAY	HARRAY	NEWO
4	HARRAY	GARRAY	NEW1	HARRAY

4. PROGRAM DESCRIPTIONS

Structural diagrams of the SIMLOG/TESTGN subprograms were given in Section 2. A more detailed description of each subprogram is given in this section in the form of a functional description. Flowcharts are included for selected routines. Additional detail is given in the annotated listings found in the appendix.

SIM1 AND SIMA

The function of these programs is to input the circuit description and simulation requirements and to build the database needed by SIM2 (SIMB). The programs consist of a main routine and four subroutines.

Main Routine

The function of the main routine of SIM1 and SIMA is to open and close files, to initialize parameters and arrays, to request mode selections, to call appropriate subroutines, and to chain to the SIM2 and SIMB programs when all other functions have been completed. A flowchart of the routine is given in Figure 14.

Keyboard Circuit-Description Subroutine

The purpose of this subroutine is to input circuit description information from the keyboard and to load the N(N$), P, I(I$), and Q(Q$) arrays.

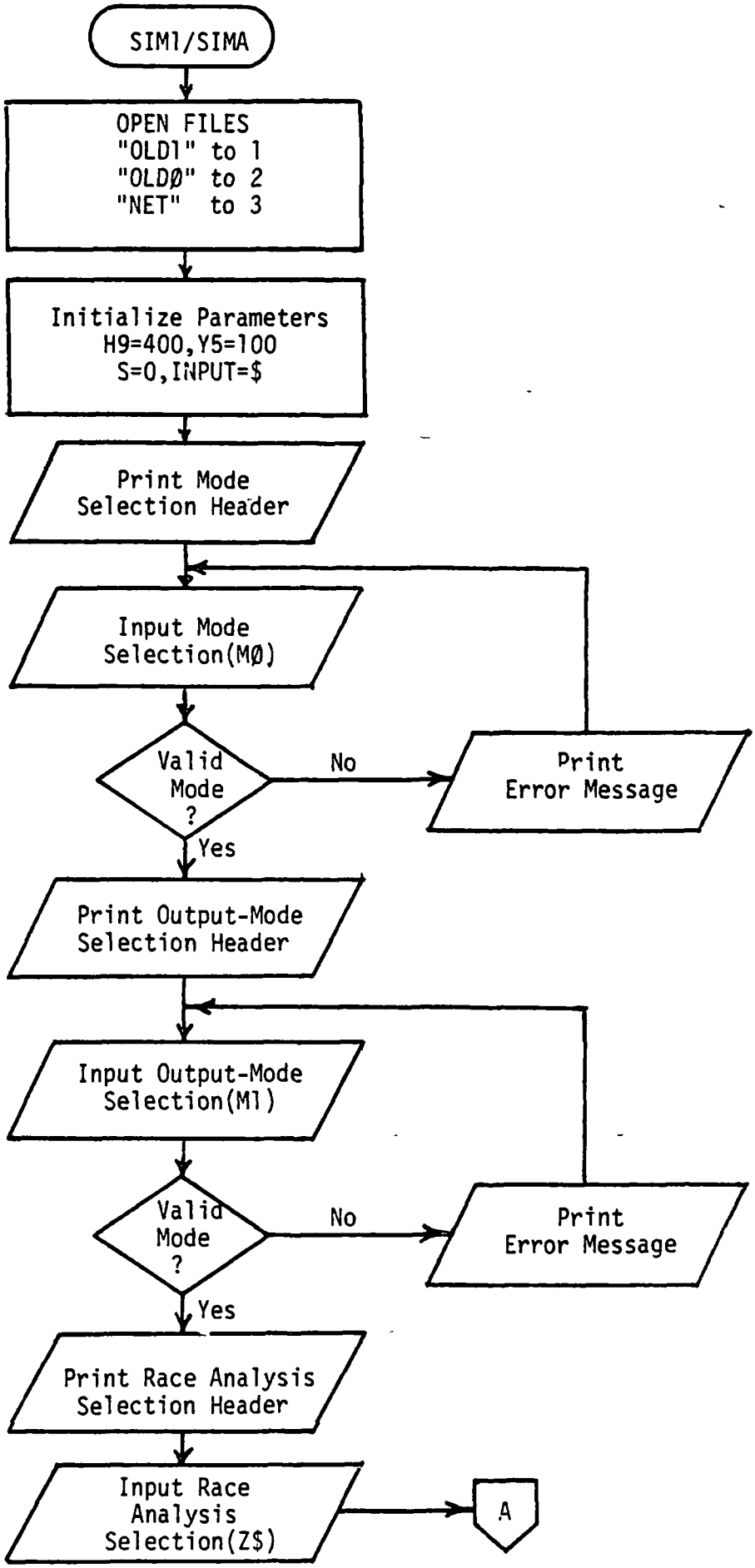


Figure 14. SIM1 and SIMA Flowchart.

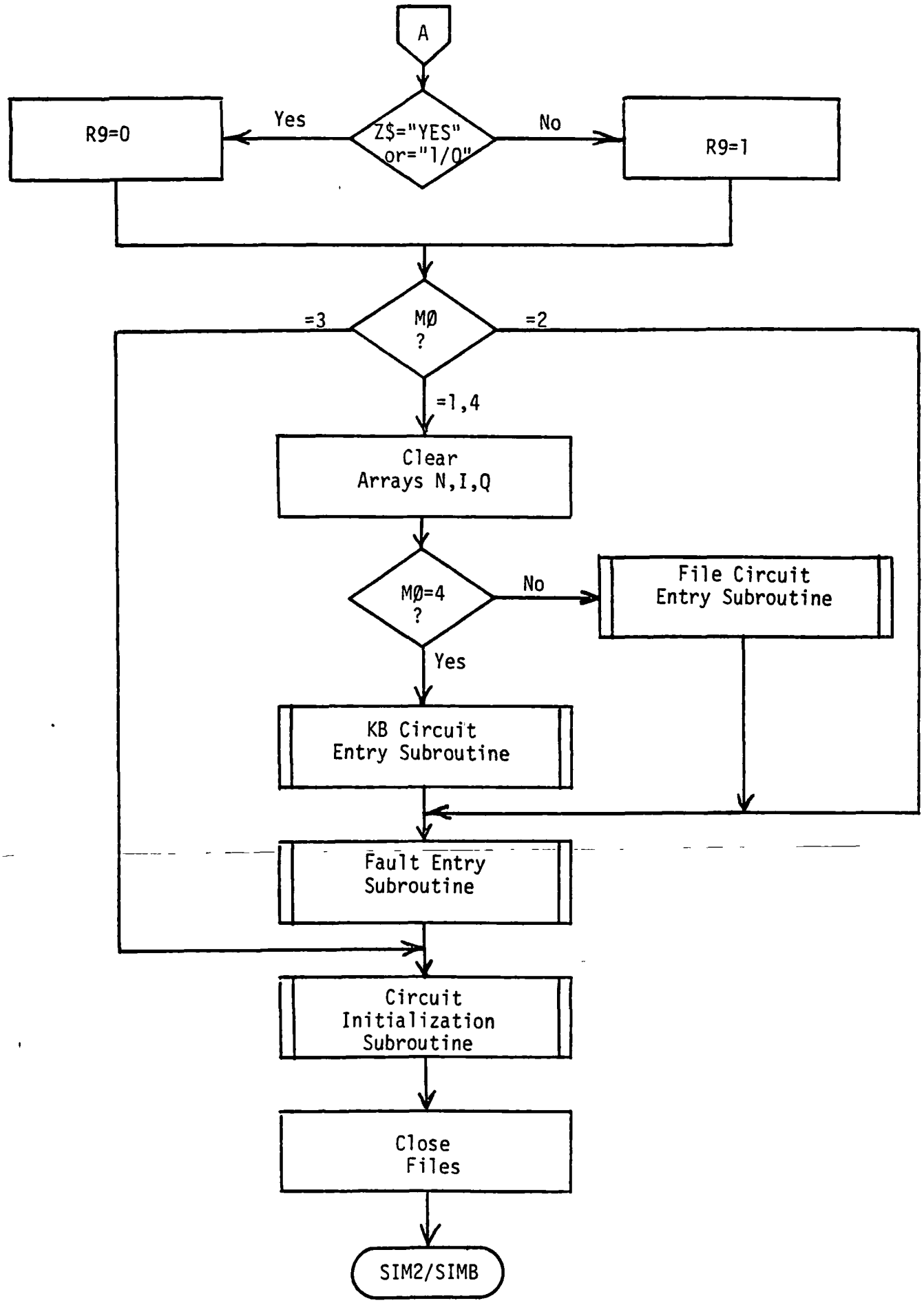


Figure 14 (con't).

Keyboard Fault-Description Subroutine

This subroutine inputs fault descriptions from the keyboard and modifies the appropriate entries of the N(N$) and I(I$) arrays.

Keyboard Starting-State Subroutine

This subroutine inputs starting-state information from the keyboard and loads the starting-state description in files OLD0 and OLD1.

File Circuit-Description Subroutine

The function of this subroutine is to input circuit description information from the file NET (NET.DAT).

SIM2

The function of SIM2 is to perform the actual simulation computations for the circuit whose description has been entered in the database by SIM1. The program consists of a main routine and sixteen subroutines.

Main Routine

This routine functions to open and close files, to initialize parameters, to input the input-time and the ripple-time limits, to initialize and increment the input-time and ripple-time counters, to call appropriate subroutines, and to chain other subprograms. See Figure 15 for a flowchart.

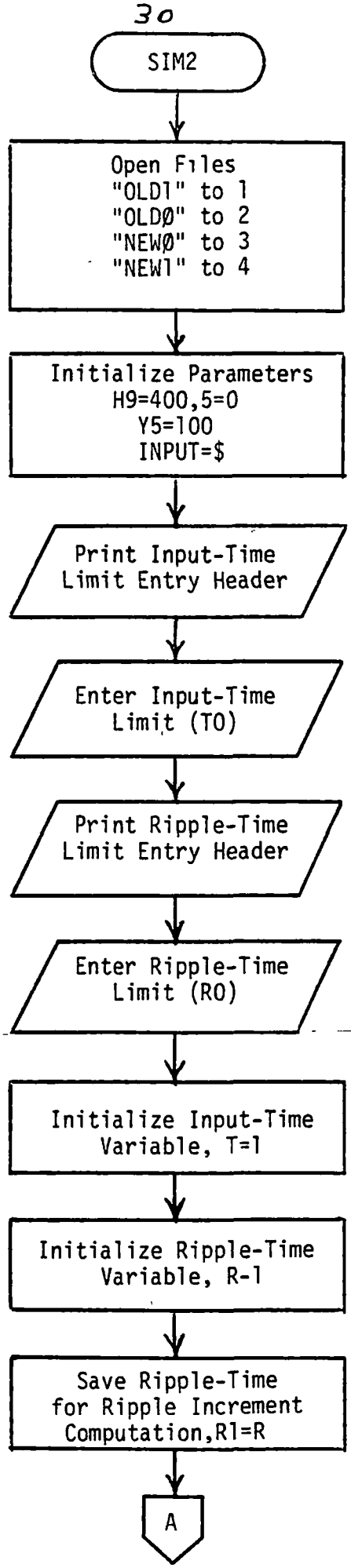


Figure 15. SIM2 Flowchart.

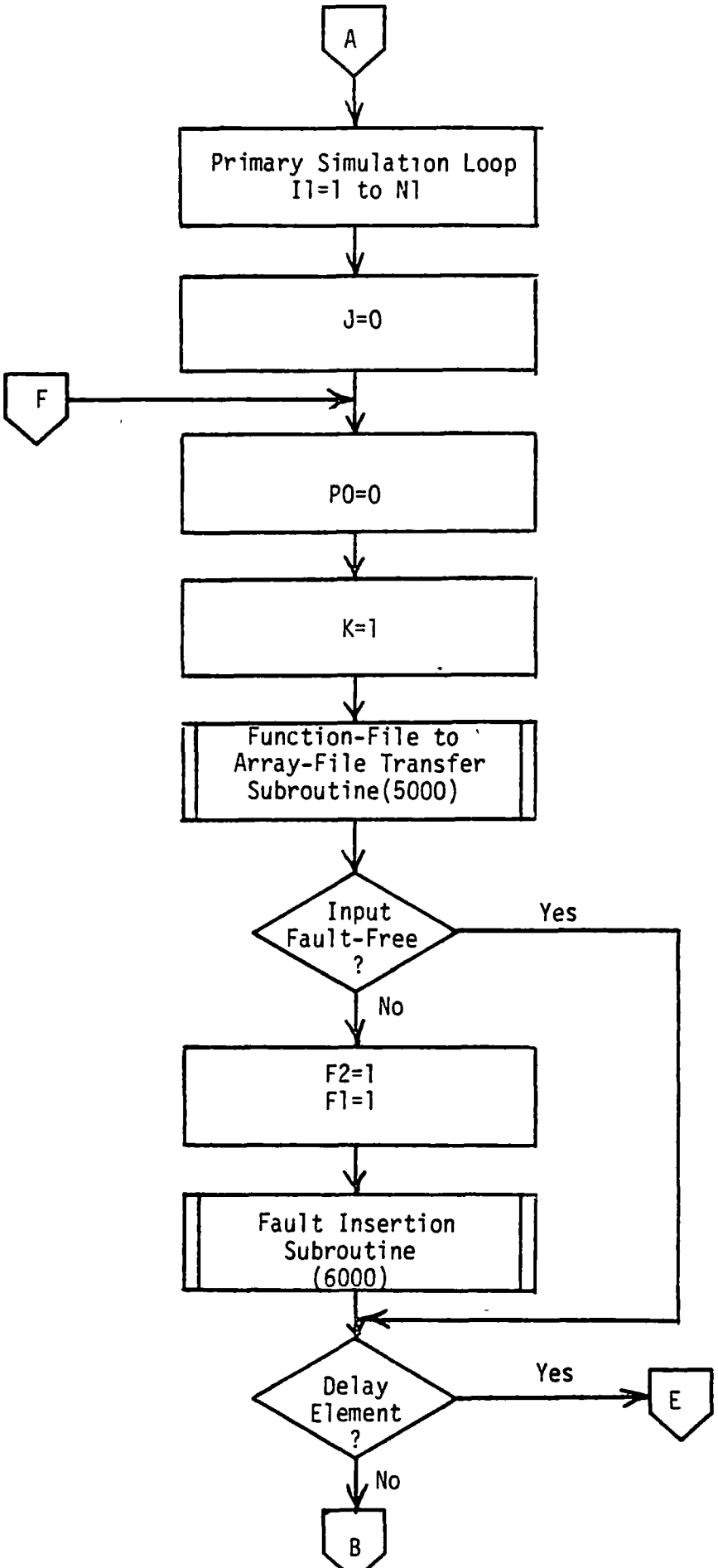


Figure 15 (cont't)

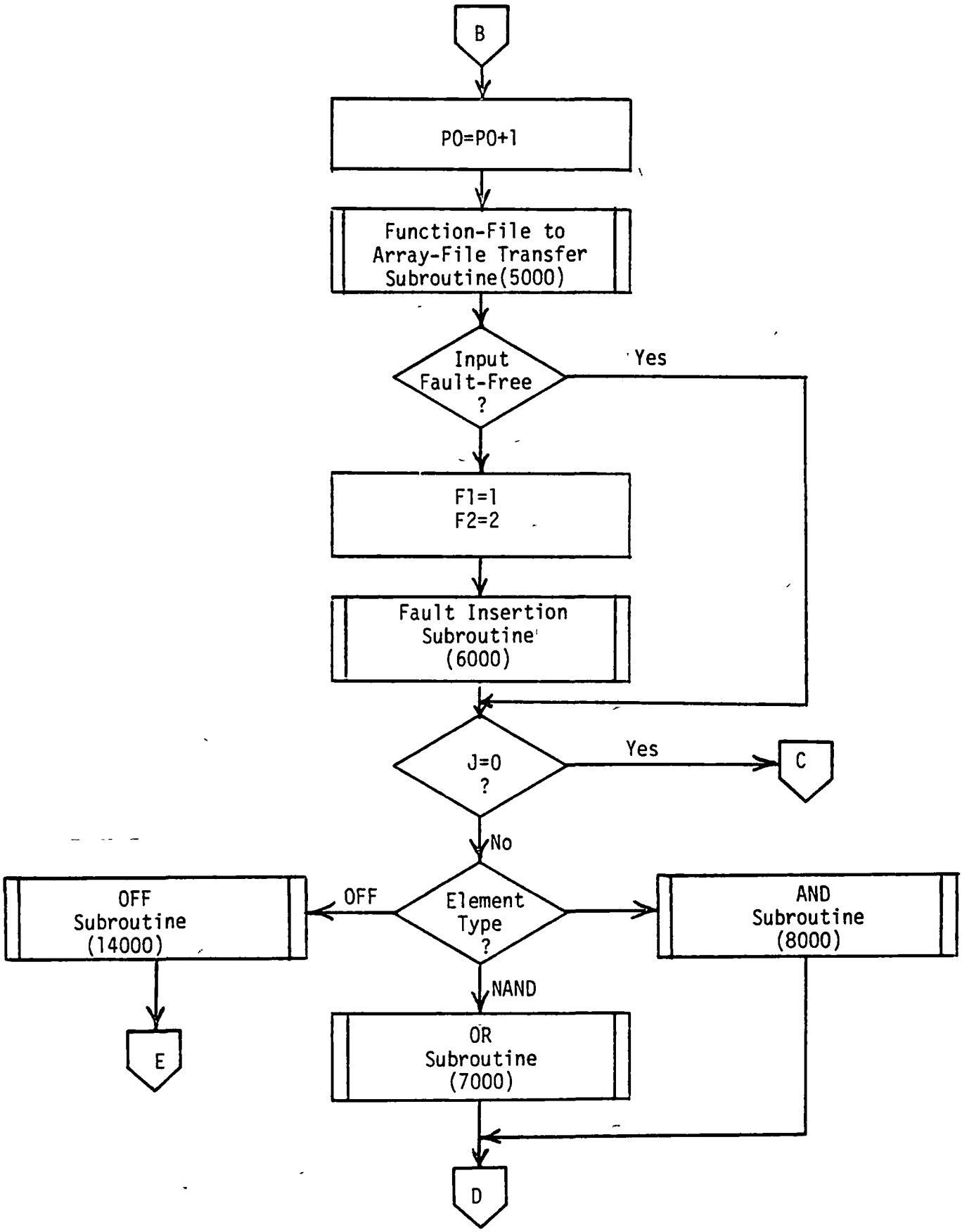


Figure 15 (con't).

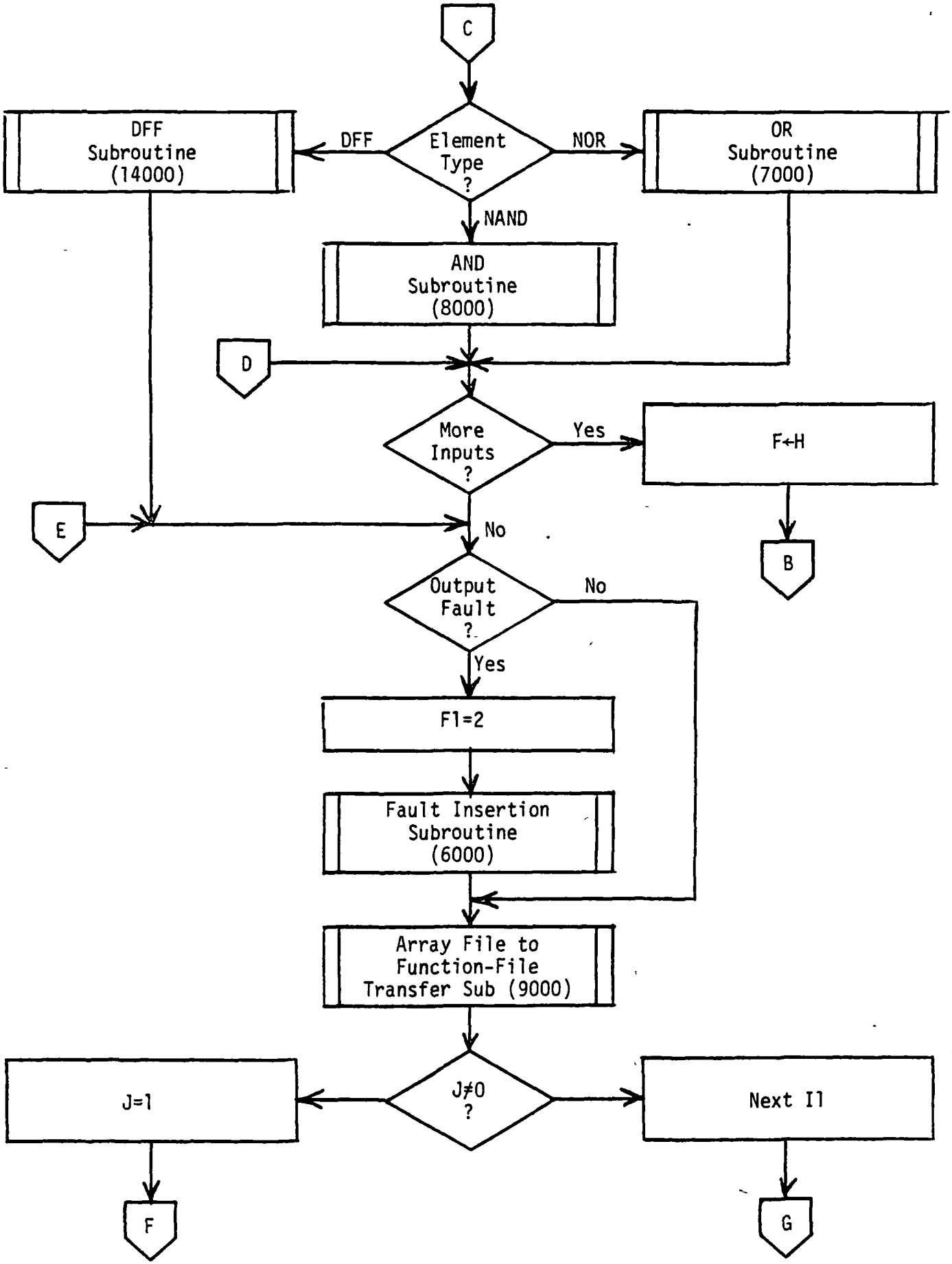


Figure 15 (con't).

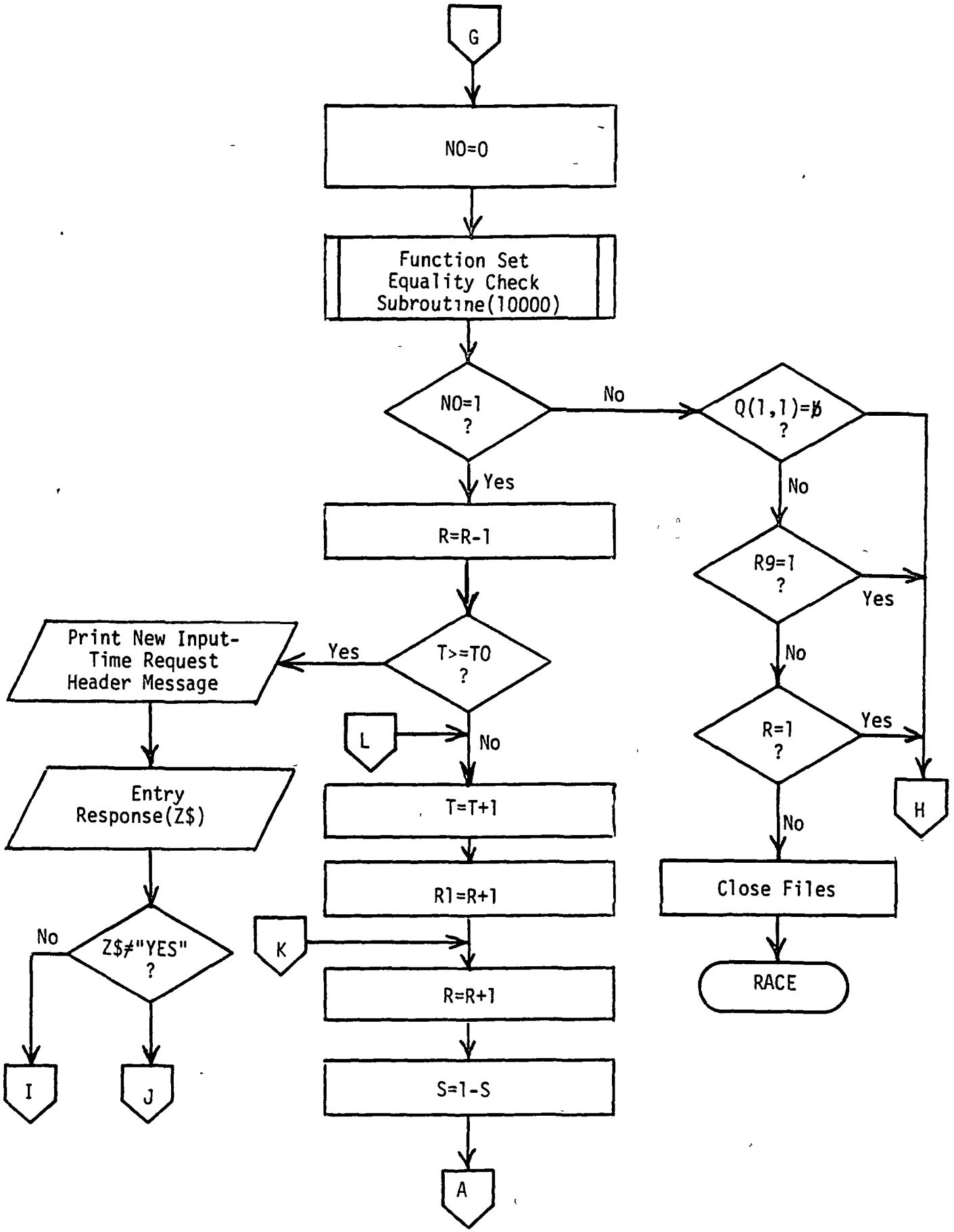


Figure 15 (con't).

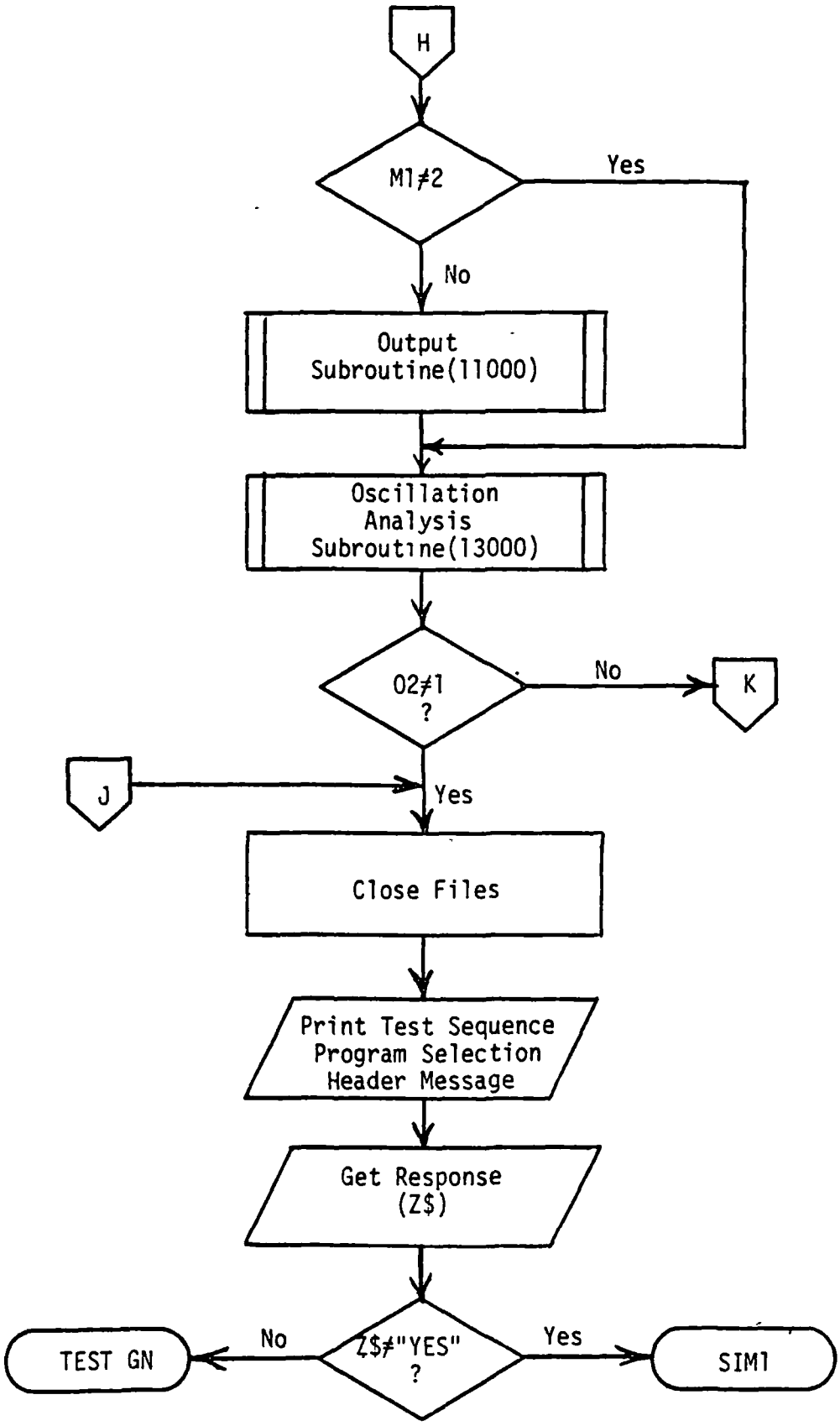


Figure 15 (con't).

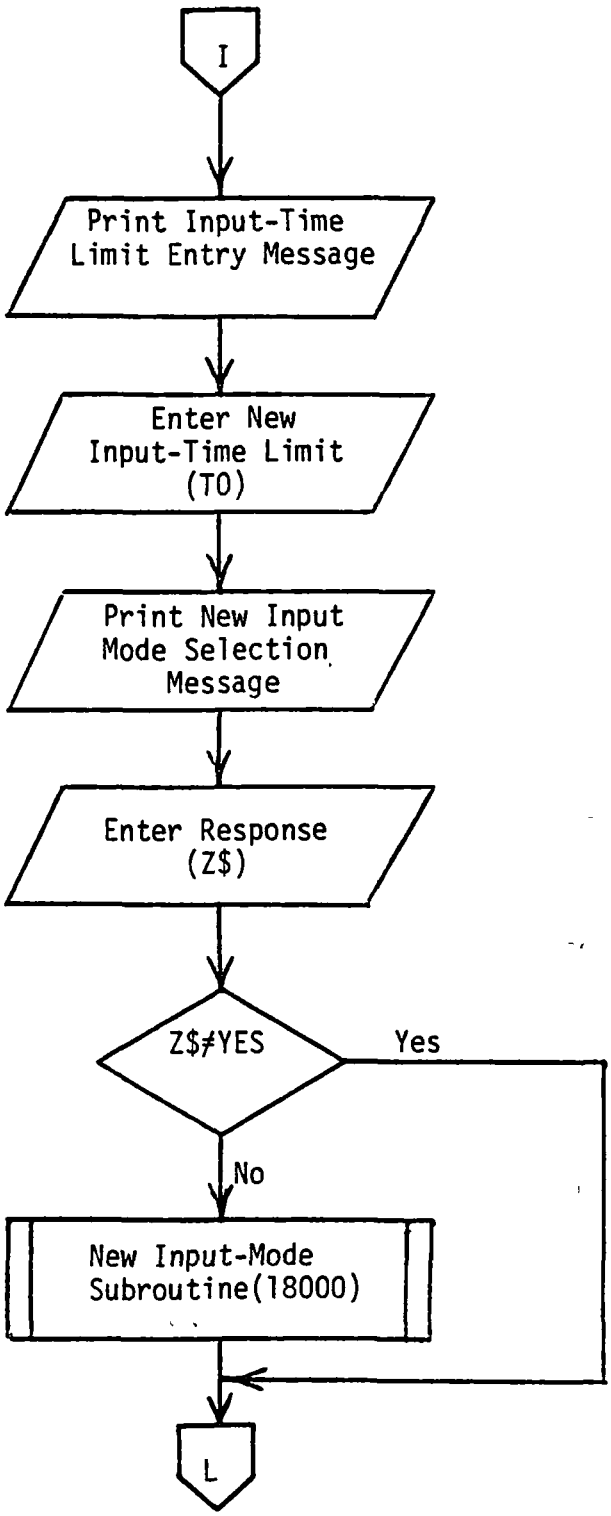


Figure 15 (con't.)

Function-File to Array-File Subroutine

The purpose of this subroutine is to copy the description of a function from a specified file record to a specified array-file (FARRAY or GARRAY). During the copy process the format of the function is changed from a single string to a list of substrings of the original string. Each substring is a product term of the function. Removal of the function name is performed.

Fault/Constant-Value Insertion Subroutine

This subroutine performs the necessary computations to effect the insertion of faults or constant input values in the circuit being simulated.

Mini-AND Subroutine

This subroutine performs the logical AND of a fault-variable with a fault-free function as part of the fault insertion procedure.

Mini-OR Subroutine

This subroutine performs the logical OR of a fault-variable and a fault-free function as part of the fault insertion procedure.

OR Subroutine

The function of this subroutine is to perform the logical OR of two functions. Operands are found in array-files FARRAY and GARRAY.

The resultant function is placed in array-file HARRAY.

Term Sorting Subroutines

This subroutine rearranges the product terms stored in HARRAY by alphanumeric order and by length.

Product Term Disassembly Subroutine

A single product term (X$) is disassembled into its literal components by this subroutine.

AND Subroutine

This subroutine performs the logical AND of two functions. Operands must be placed in FARRAY and GARRAY. The resultant is placed in HARRAY.

Zero Product-Term Detection Subroutine

This subroutine detects zero product-terms that may be produced by an AND operation.

Literal Sort Subroutine

The purpose of this subroutine is to arrange the literals in a product term in alphanumeric order and to remove duplicate literals.

Array-File to Function-File Subroutine

The function of this subroutine is to assemble the list of product terms in array-file HARRAY into a string format and to place the resultant string in the specified record of file OLD0, OLD1, NEW0, or

NEW1. In string form, the function consists of a function name and time header followed by an equal sign followed by the product terms separated by plus signs.

Function Set Comparison Subroutine

This subroutine compares the functions at ripple-time r-1 to the corresponding functions at ripple-time r in order to determine if the two sets of functions are identical.

Output Subroutine

This subroutine outputs a header giving input-time and ripple-time. The header is followed by a list of function pairs for each net in the circuit.

Oscillation Detection Subroutine

This subroutine checks the increment of ripple-time change since the last input-time change against the ripple-time limit in order to identify a possible oscillation condition.

Function Minimization Subroutine

The purpose of this subroutine is to remove redundant terms from a function. Terms are redundant in this context if they are covered by another term.

Input-Mode Change Subroutine

This subroutine permits the user to change input mode from fixed input mode to variable input mode or vice-versa.

RACE

The RACE subprogram performs the race analysis procedure and is chained from SIM2. A main-routine and four subroutines constitute the subprogram. The AND subroutine is the same as that used in SIM2 and will not be described again.

Main Routine

This routine opens and closes files, initializes parameters, examines functions corresponding to cross-coupled gates, calls the appropriate subroutines, and chains to SIM2 when all other operations are complete. See Figure 16 for a flowchart.

Name Removal Subroutine

This subroutine removes the name field from the string form of a function to be examined.

String to Matrix Subroutine

This subroutine converts the string form of a function to a list of product terms.

Matrix to String Subroutine

This subroutine performs the inverse of the above subroutine.

SIMB

This subprogram combines the functions of SIM1 and RACE in one program. No further discussion in necessary.

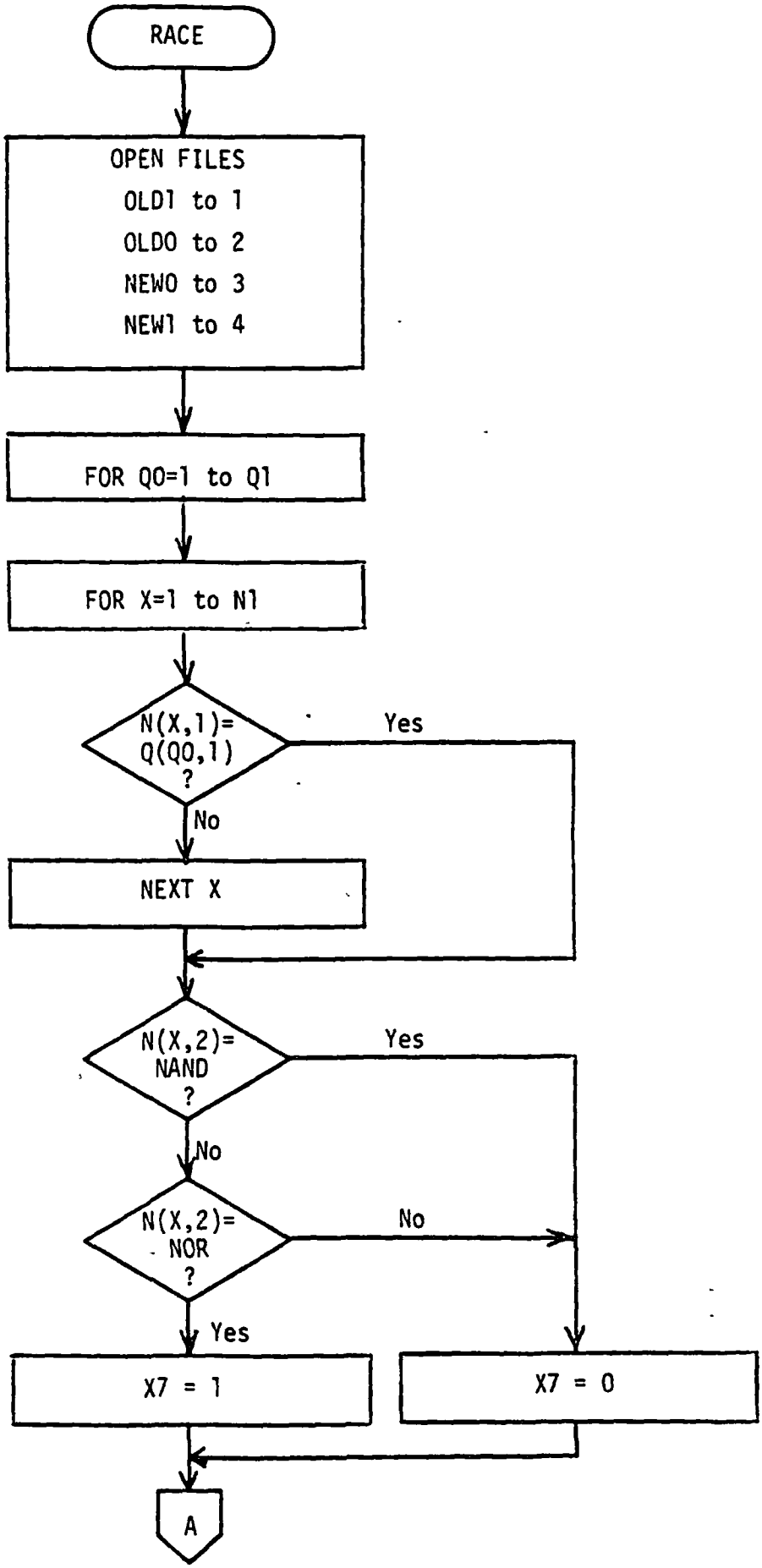


Figure 16. RACE Flowchart.

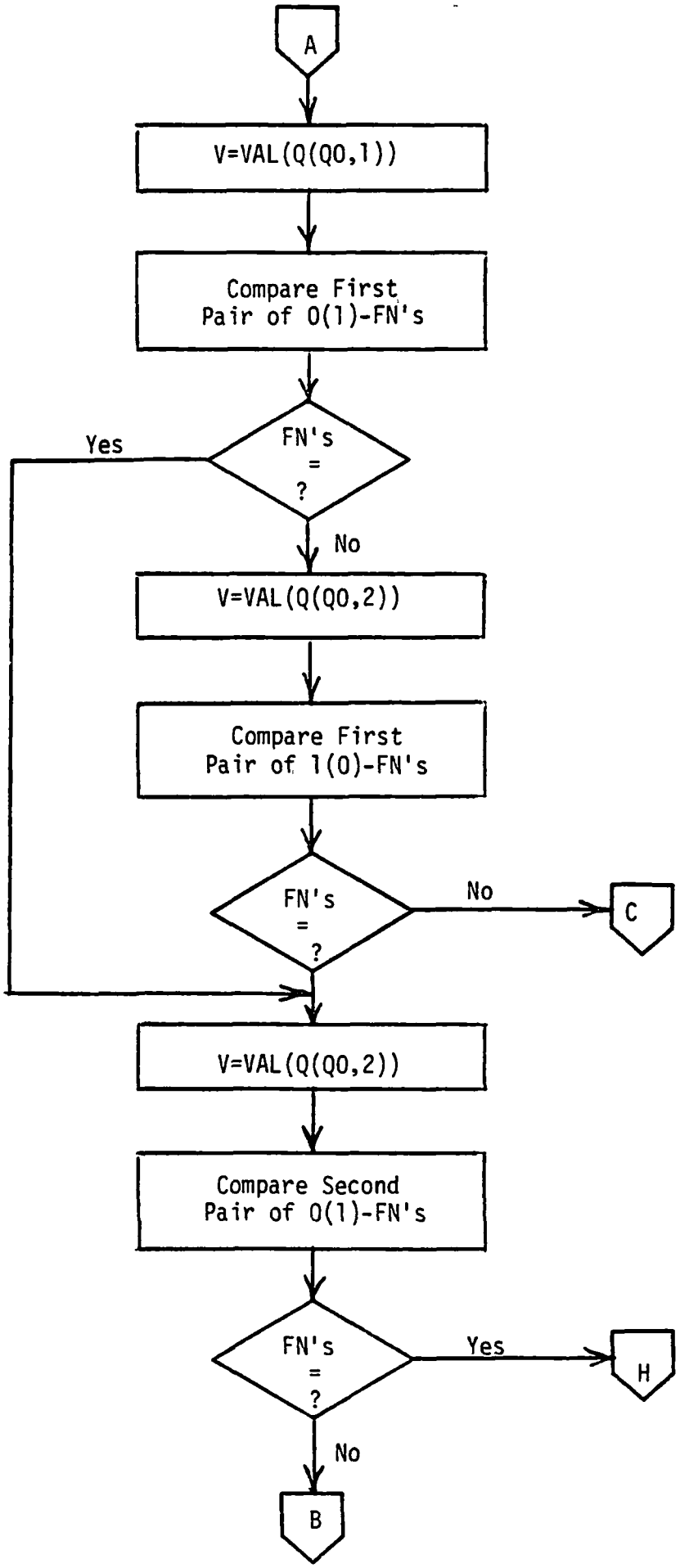


Figure 16 (con't).

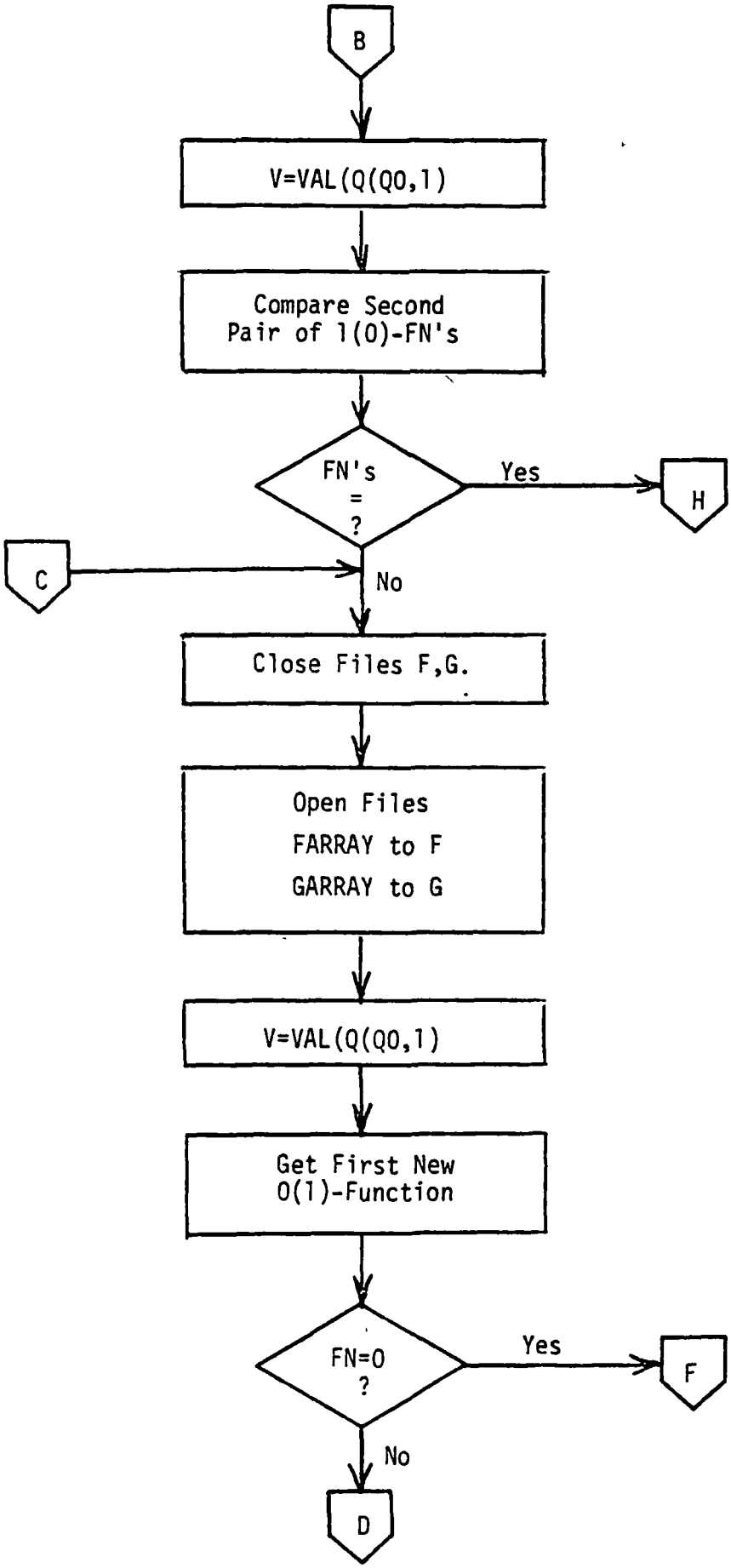


Figure 16 (con't).

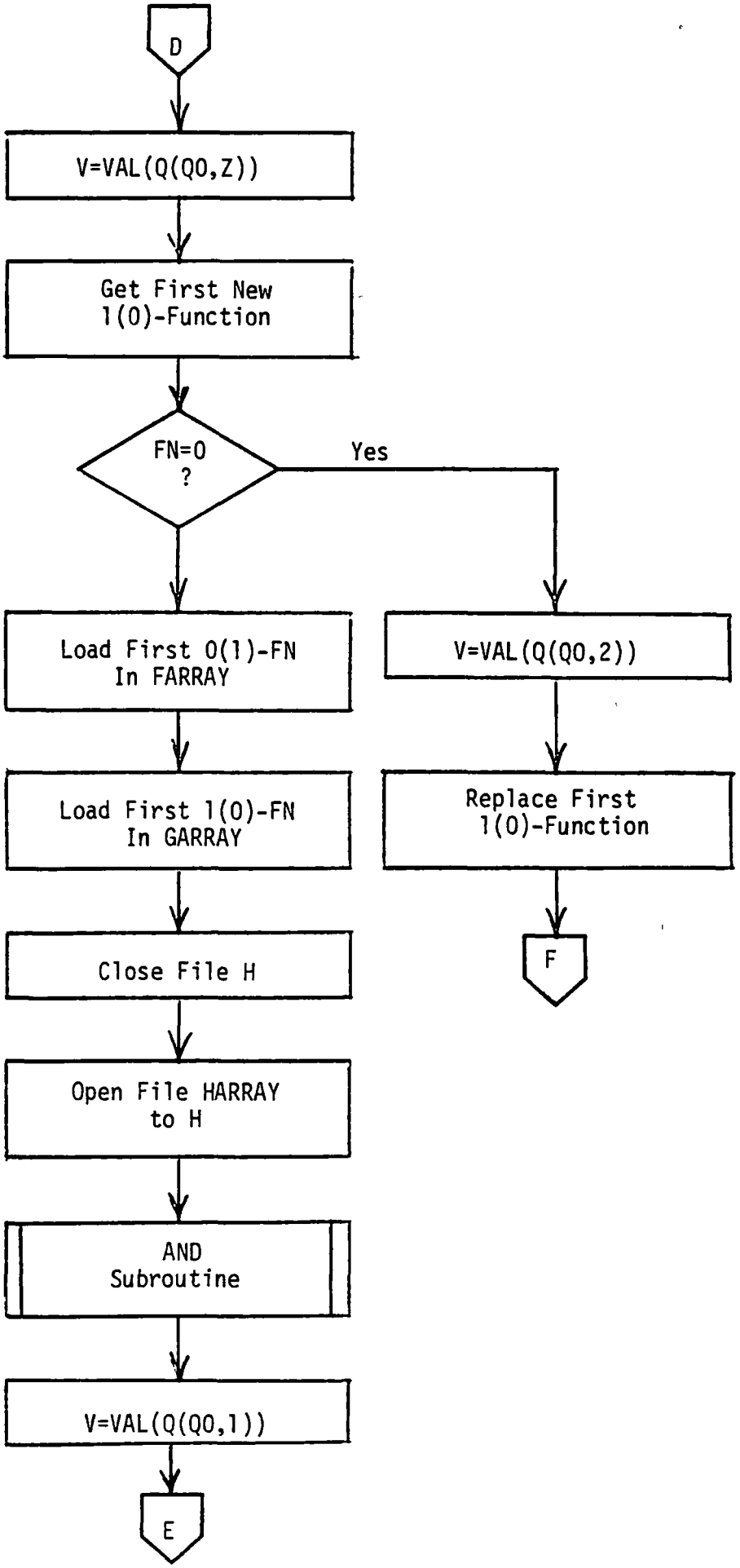


Figure 16 (con't).

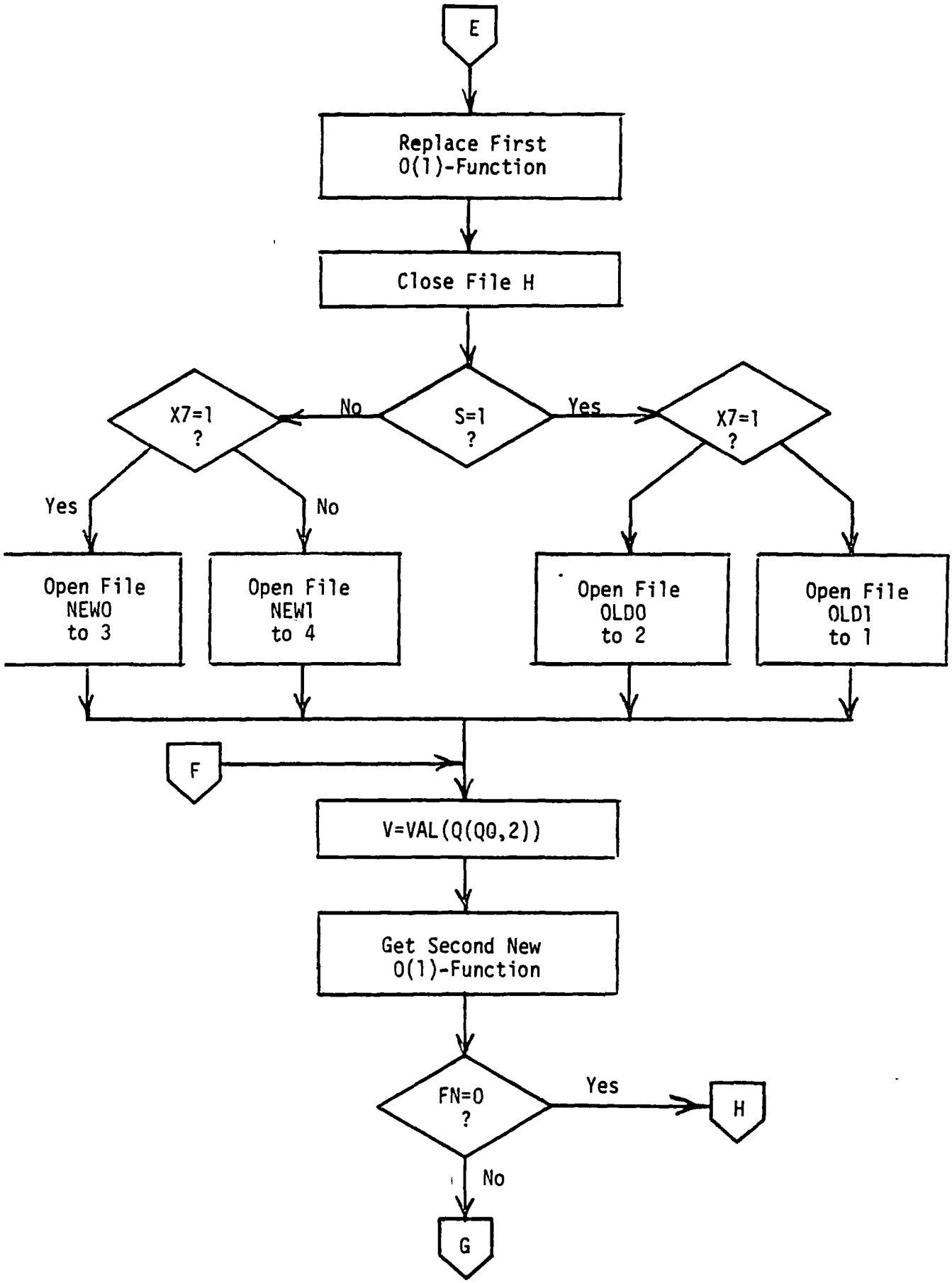


Figure 16 (con't).

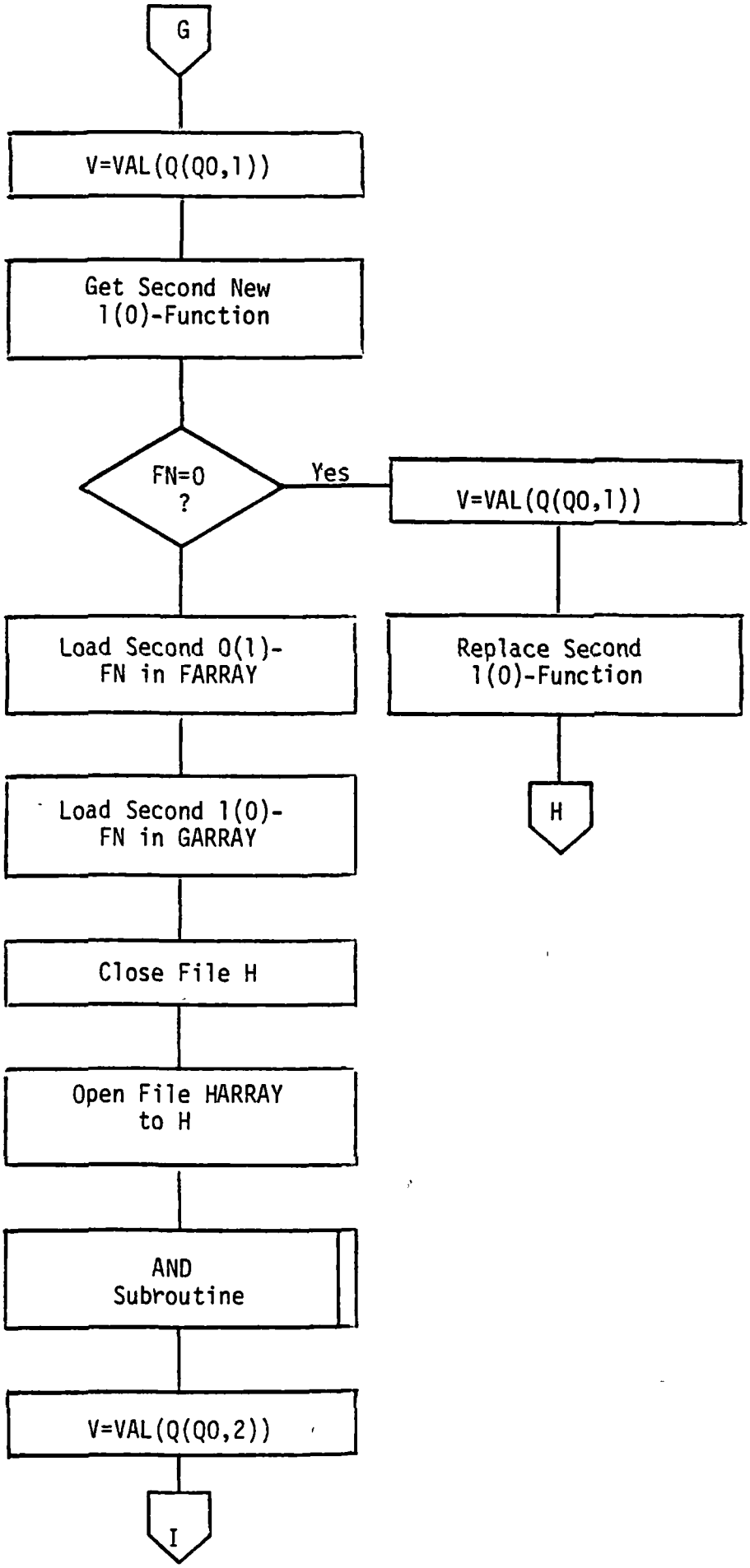


Figure 16 (con't).

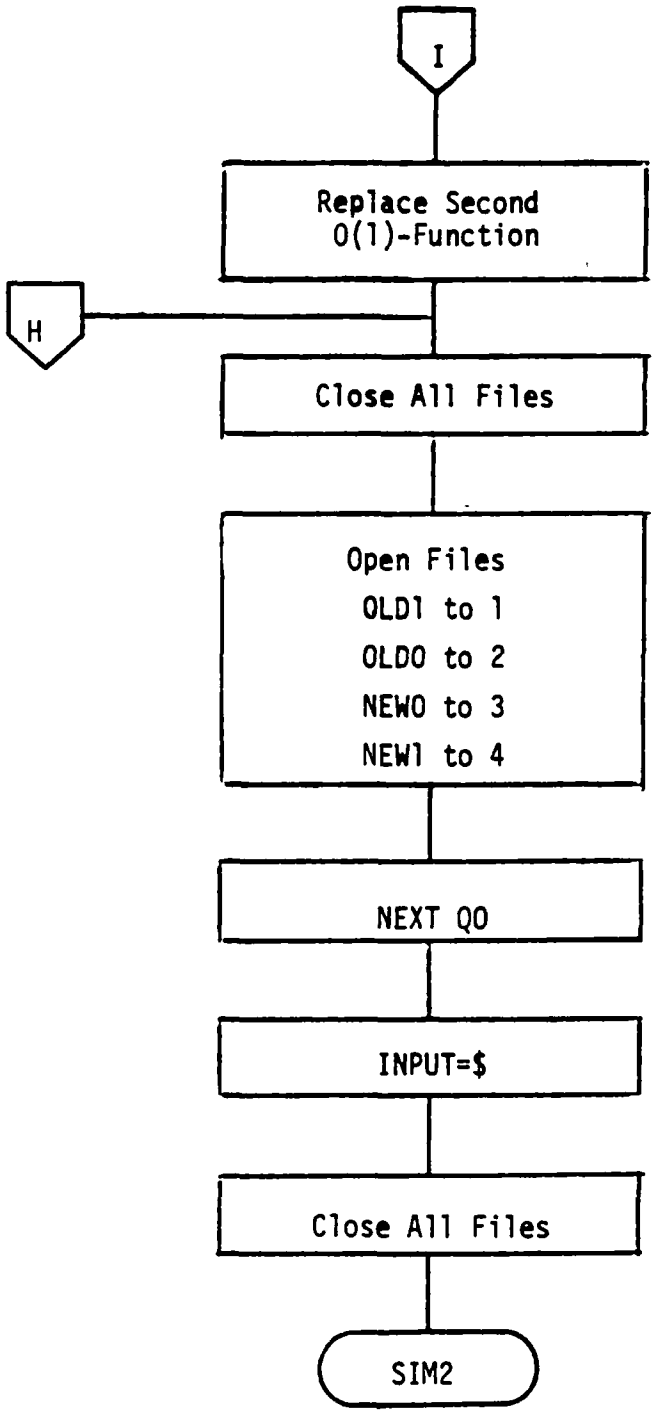


Figure 16 (con't).

TESTGN

The generation of test sequences from SIMLOG simulations is the function of this subprogram. It consists of a main routine and fourteen subroutines. Most of the subroutines are identical to ones used in SIM2 and will not be described again.

Main Routine

The function of the main routine is to open and close files, to initialize parameters, to input the test generation mode selection, to construct circuit input and output lists, to select input-output pairs for processing by subroutines, to call appropriate subroutines, and to chain to SIM1. See Figure 17 for a flowchart.

Test Function Generation Subroutine

This subroutine generates the test function for a specified input net, output net, and input-time, or for a specified fault condition and output net. See Figure 18 for a flowchart.

Output Function Retrieval Subroutine

This subroutine sets switches as needed by the function-file to array-file subroutine for transferring the appropriate function.

Test Function Storage Subroutine

Test functions are placed in a file by this subroutine after each is generated.

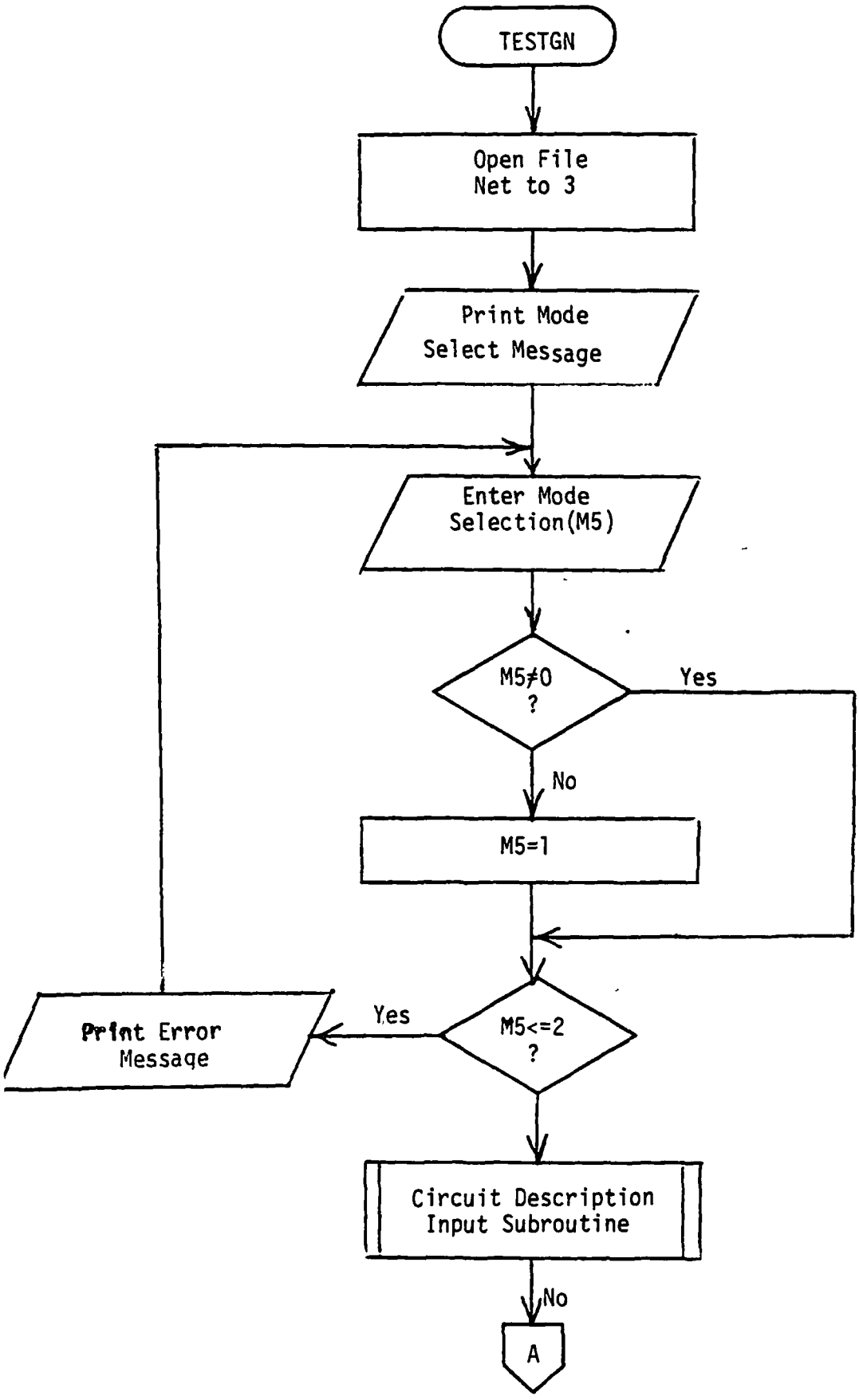


Figure 17. TESTGN Flowchart.

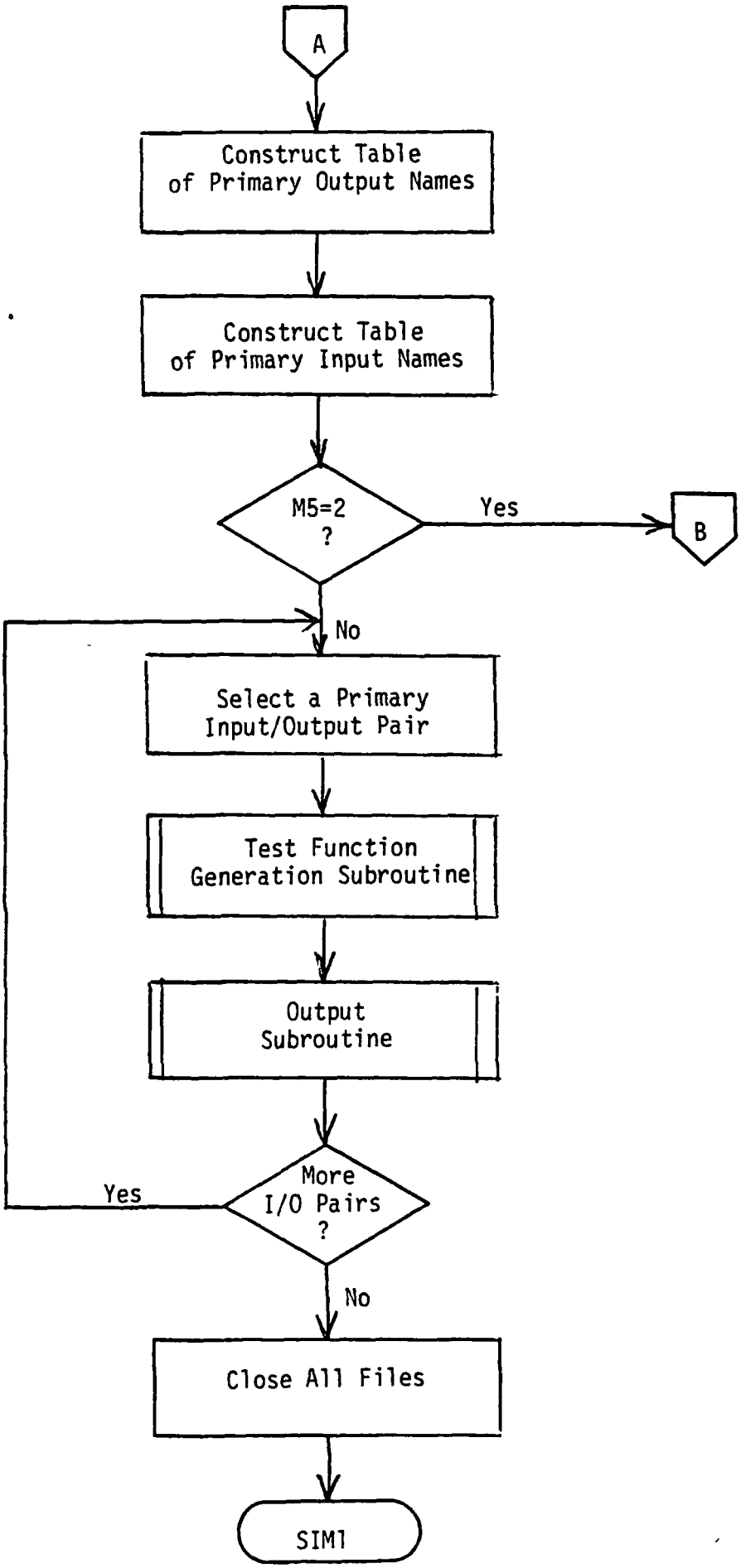


Figure 17 (con't).

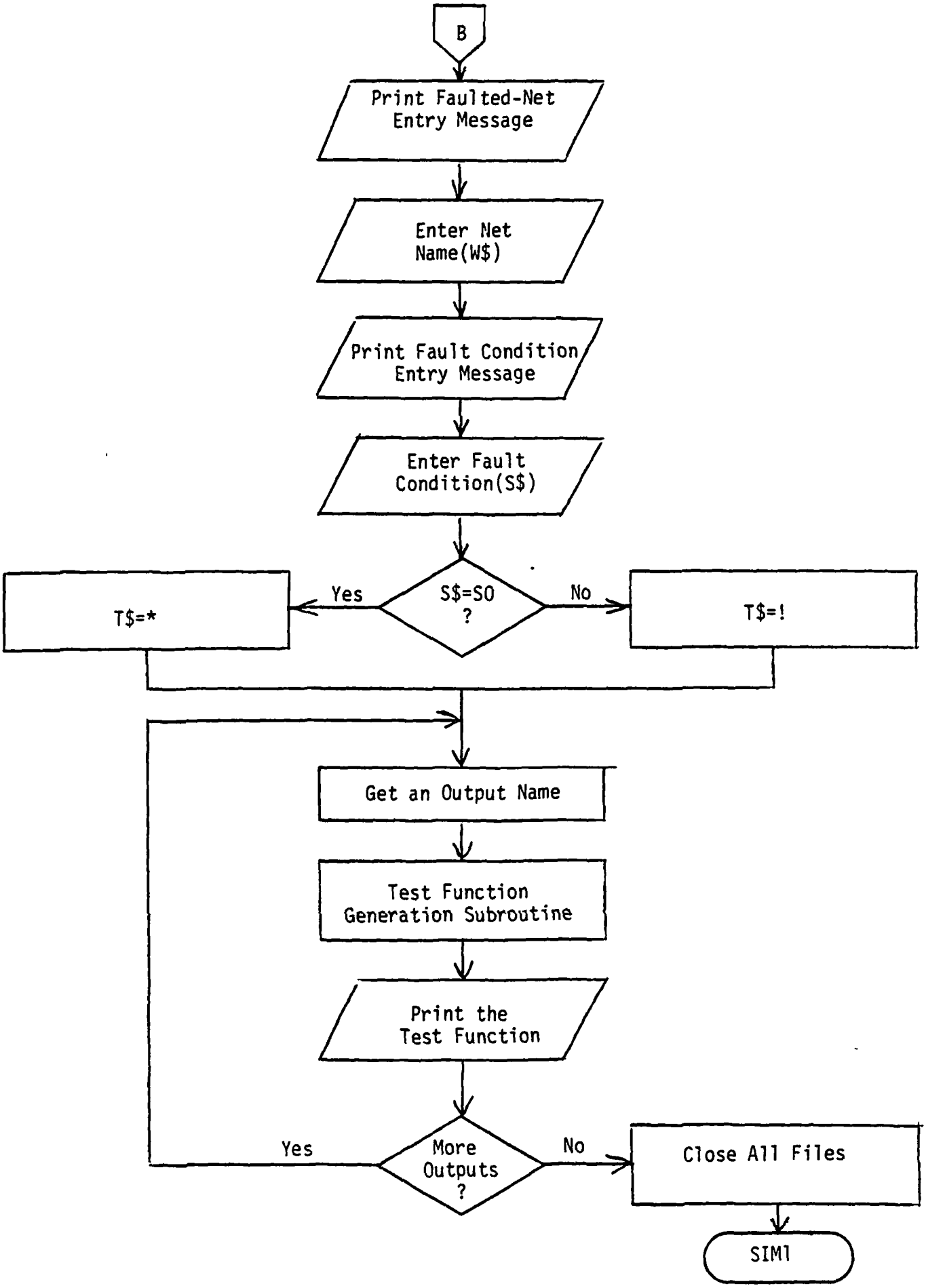


Figure 17 (con't).

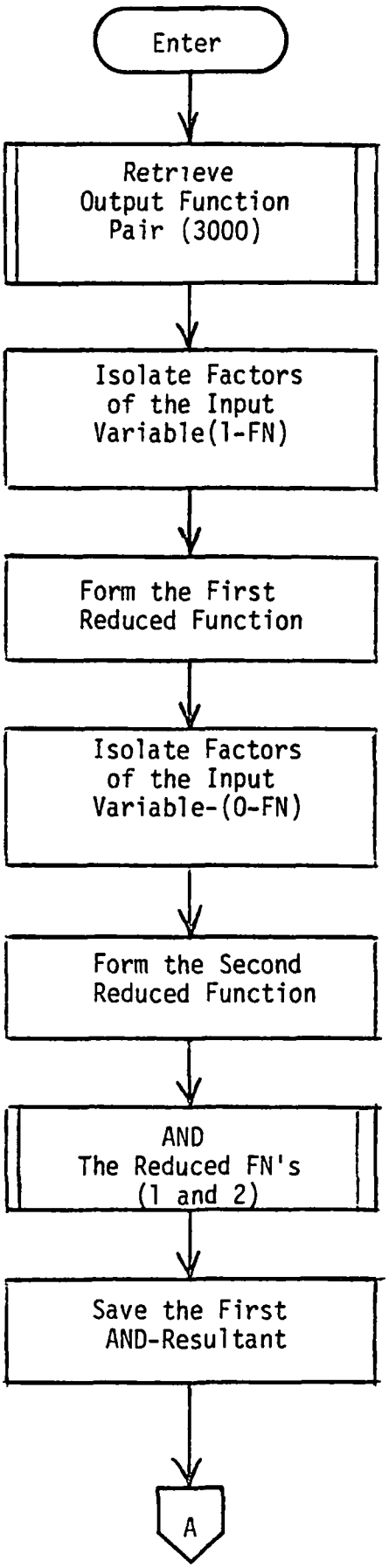


Figure 18. Test Function Computation Routine Flowchart

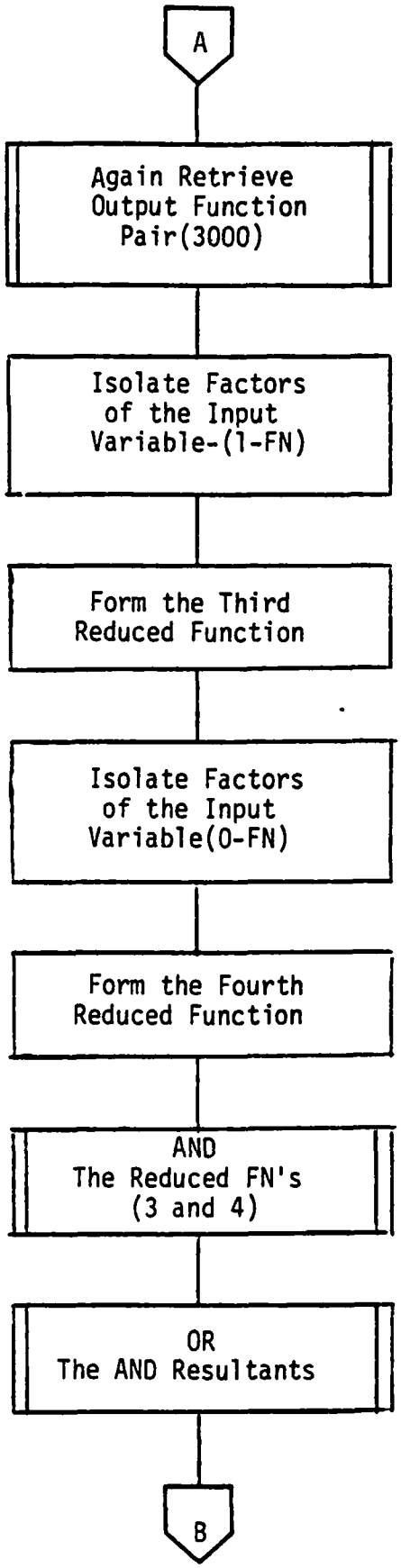


Figure 18. (con't)

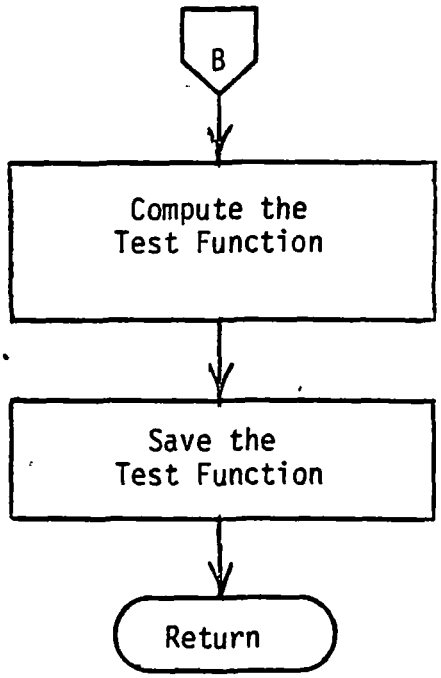


Figure 18. (con't)

Output Subroutine

A header including the output name, the input name, and the time tag are printed if mode 1 is chosen. If mode 2 was chosen, the output name and the fault condition are printed. The corresponding test function is then printed.

5. PROGRAM MODIFICATION GUIDELINES

Suggestions for program modifications are given in this section. The first subsection contains guidelines for modifying program parameters that affect the size of circuit that can be handled by the programs. The second subsection contains suggestions for changes to the program that should improve its performance characteristics.

PARAMETER SETTINGS

The size circuit that can be handled by the system is determined by several parameters that will now be discussed. Dimensions of arrays N(N$), P, I(I$), and Q(Q$) directly influence the maximum circuit size that can be handled.

The number of rows in arrays N(N$) and P must equal or exceed the number of circuit elements (a D flip-flop counts as two elements). The number of rows in array I(I$) must equal or exceed the total number of element inputs in the circuit. Therefore, the number of rows in I(I$) must be at least double the number of rows in N(N$) or P. Array Q(Q$) must have at least as many rows as there are cross-coupled gate pairs. A Q(Q$) array with one fourth the number of rows as array N(N$) is recommended.

Parameter H9 is used as a FOR-NEXT loop limit and should be set equal to or greater than the number of rows in I(I$). In the Sigma 5 version, parameter Y5 sets the maximum number of product terms that can be contained in a function.

SUGGESTIONS FOR IMPROVEMENT

Significant improvements in performance (speed) and memory requirements could be achieved if the programs were rewritten in a language that could be compiled into machine code. Another possibility is to develop a BASIC compiler for the Sigma 5. Such a compiler is now available on some PDP 11 installations.

Modifications to the existing code can also be effective in providing substantial improvements. One such possibility is to develop a uniform internal code for representing nets and inputs. Such a code would improve performance and memory requirements. This feature would also permit a more flexible net labeling scheme. Addition of such a feature would require translation routines to map the user selected net labels to the internal code and vice-versa. A symbol table would also be required.

Another desirable modification is to improve the function minimization routine so that a more extensive minimization is performed. Recognition of terms of the form X + X- is the most needed improvement in this regard.

6. REFERENCES

1. B. D. Carroll, "Test Pattern Generation," Final Report-Vol. 2, NAS8-31572, Electrical Engineering Department, Auburn University, Auburn, Alabama, September 14, 1979.

2. B. D. Carroll, "SIMLOG/TESTGN User's Guide," Final Report-Vol. 2, Addendum 1, NAS8-31572, Electrical Engineering Department, Auburn University, Auburn, Alabama, September 14, 1979.

APPENDIX

SIMLOG/TESTGN

ANNOTATED LISTINGS

SIGMA 5 VERSION

```
10:10 OCT 11,'79 DC/SIM1DOC.BCARROLL

1*       SSSSS    III    M   M    LL        00000    GGGGG
2*       S          I    MM MM    L         0   0    G
3*       SSSSS      I    M M M    L         0   0    G  GG
4*           S      I    M   M    L         0   0    G   G
5*       SSSSS    III    M   M    LLLLL     00000    GGGGG
6*
7*             PROGRAM:          SIM1DOC
8*             VERSION:          SIGMA 5
9*             REVISION:         ORIGINAL
10*            DATE:             10/10/79
11*            PROGRAMMER:       B. D. CARROLL
12*
13*************************************************************************
14*
15*             CONTRACT SUPPORT
16*************************************************************************
17*
18*       THIS PROGRAM WAS DEVELOPED FOR NASA MARSHALL FLIGHT
19*       CENTER UNDER CONTRACT NAS8-31572.
20*
21*************************************************************************
22*
23*             MODIFICATION HISTORY
24*
25*************************************************************************
26*
59*
100************************************************************************
101*
102*             PROGRAM DESCRIPTION
103*
104************************************************************************
105*
106*       SIMLOG IS A PROGRAM FOR SIMULATING LOGIC CIRCUITS.
107*       BOTH COMBINATIONAL AND SEQUENTIAL CIRCUITS CAN BE
108*       SIMULATED.  CIRCUITS CAN BE SIMULATED FAULT-FREE OR
109*       WITH SINGLE OR MULTIPLE STUCK TYPE FAULTS.
110*       LOGIC ELEMENTS ACCOMMODATED BY THE SIMULATOR ARE
111*       NAND GATES, NOR GATES, AND DELAY ELEMENTS.
112*       OTHER ELEMENT TYPES SUCH AS D FLIP-FLOPS WILL BE ADDED.
113*
114*       TWO VERSIONS OF SIMLOG HAVE BEEN WRITTEN.
115*       ONE VERSION HAS BEEN WRITTEN IN BASIC-PLUS FOR
116*       EXECUTION ON A PDP11/40 RSTS/E SYSTEM.
117*       ANOTHER VERSION HAS BEEN WRITTEN IN XEROX BASIC
118*       FOR EXECUTION ON A SIGMA 5 CPV SYSTEM.
119*
120*       THE PDP11/40 VERSION IS PARTITIONED INTO TWO
121*       SUBPROGRAMS, SIMA AND SIMB.  THIS PARTITIONING WAS
122*       NECESSARY DUE TO THE LARGE MEMORY REQUIREMENTS OF
```

```
123*     THE PROGRAM.  VIRTUAL ARRAYS ARE USED FOR ALL LARGE
124*     ARRAY STORAGE.
125*
126*     THE SIGMA 5 VERSION IS PARTITIONED INTO THREE
127*     SUBPROGRAMS, SIM1, SIM2, AND RACE.  THIS DEGREE OF
128*     PARTITIONING WAS NECESSARY SINCE VIRTUAL ARRAYS ARE
129*     NOT AVAILABLE ON THE SIGMA 5.
130*
131*     THE FUNCTION OF SIM1 IS TO INPUT ALL DATA NEEDED
132*     CONCERNING CIRCUIT DESCRIPTION, FAULT DESCRIPTION,
133*     INITIAL CONDITIONS, AND SIMULATION MODES.
134*
135*     THE FUNCTION OF SIM2 IS TO PERFORM THE ACTUAL
136*     SIMULATION COMPUTATIONS WITH THE EXCEPTION OF
137*     RACE ANALYSIS AND TO OUTPUT THE SIMULATION RESULTS.
138*
139*     THE FUNCTION OF RACE IS TO PERFORM THE RACE
140*     ANALYSIS COMPUTATIONS.
141*
142*
300*****************************************************************
301*
302*          I/O CHANNELS
303*
304*****************************************************************
305*
306*
309*     CHANNEL          USAGE
310*
311*
312*          1           FILE OF OLD ONE-FUNCTIONS
313*
314*          2           FILE OF OLD ZERO-FUNCTIONS
315*
316*          3           CIRCUIT DESCRIPTION FILE
317*
318*
400*****************************************************************
401*
402*          VARIABLE DEFINITIONS
403*
404*****************************************************************
405*
409*     VARIABLE         DEFINITION
410*
411*          A           AN INPUT VARIABLE USED FOR THE STARTING-STATE
412*                      MODE NUMBER.
413*
414*          AI          AN INPUT VARIABLE USED FOR A SUBMODE NUMBER
415*                      DURING THE STARTING-STATE INPUT.
416*
417*          B$          A STRING VARIABLE USED AS A TEMPORARY VARIABLE
```

```
418*            AND AS AN INPUT VARIABLE.
419*
420*      C$    A STRING VARIABLE USED AS A TEMPORARY VARIABLE
421*            AND AS AN INPUT VARIABLE.
422*
423*      H9    INDICATES THE UPPER LIMIT OF THE LOOP VARIABLE
424*            OF SOME FOR-NEXT LOOPS.  SHOULD BE SET >= THE
425*            NUMBER OF ROWS IN THE I ARRAY.
426*
427*      I     A TWO DIMENSIONAL IMPLICIT STRING ARRAY USED TO
428*            STORE CIRCUIT ELEMENT INPUT LISTS.  EACH ROW OF
429*            THE ARRAY DESCRIBES ONE INPUT VARIABLE OF ONE
430*            ELEMENT OF THE CIRCUIT BEING DESCRIBED.  THE
431*            VARIABLE NAME IS GIVEN IN COLUMN ONE.  ANY FAULT
432*            CONDITION ASSOCIATED WITH THE INPUT IS
433*            GIVEN IN COLUMN TWO.  THE THIRD COLUMN INDICATES
434*            WHETHER OR NOT THE INPUT IS A PRIMARY INPUT.
435*
440*      I0    A POINTER TO THE NEXT ROW OF THE I ARRAY TO BE
441*            LOADED DURING INPUT OF THE CIRCUIT DESCRIPTION.
442*            THE FINAL VALUE OF I0 INDICATES THE NUMBER OF
443*            ROWS OF ARRAY I THAT ARE USED.
445*
450*      I1    A POINTER TO THE NEXT FILE RECORD TO BE ACCESSED.
451*            USED IN SUBR 2000 WHEN LOADING THE NET FILE.
452*
460*      L     REPRESENTS THE LENGTH OF A STRING.  VALUE IS
461*            ESTABLISHED BY THE LEN FUNCTION.
462*
470*      M0    SIMULATION MODE INPUT VARIABLE.
471*            LOADED IN MAIN PROGRAM.
472*
480*      M1    OUTPUT-MODE INPUT VARIABLE.  LOADED IN MAIN PROG.
481*            M1=1: PRINT ALL EQUATION SETS.
482*            M1<>1: PRINT STABLE EQUATION SETS ONLY.
483*
490*      N     A TWO DIMENSIONAL IMPLICIT STRING ARRAY USED TO
491*            STORE THE ELEMENT DESCRIPTIONS AND INTERCONNECTIONS
492*            OF A CIRCUIT.  EACH ROW OF THE ARRAY CORRESPONDS TO
493*            ONE ELEMENT OF THE CIRCUIT.  COLUMN ONE CONTAINS
494*            THE ELEMENT OUTPUT NAME.  COLUMN TWO IS THE ELEMENT
495*            TYPE.  COLUMN THREE INDICATES THE FAULT CONDITION
496*            OF THE ELEMENT OUTPUT.  COLUMN FOUR INDICATES
497*            WHETHER OR NOT THE ELEMENT OUTPUT IS A PRIMARY
498*            OUTPUT OF THE CIRCUIT.
499*
510*      N1    A POINTER TO THE N AND P ARRAYS.  DURING INPUT OF
511*            THE CIRCUIT DESCRIPTION, N1 INDICATES THE NEXT ROW
512*            OF THE ARRAYS TO BE LOADED.
513*            AFTER CIRCUIT ENTRY IS COMPLETE, N1 INDICATES THE
514*            NUMBER OF ROWS LOADED IN N AND P.
515*
```

```
520*        P       A TWO DIMENSIONAL NUMERIC ARRAY USED TO STORE
521*                POINTERS TO THE INPUT LISTS STORED IN ARRAY I.
522*                ROW I OF P CORRESPONDS TO THE ELEMENT DESCRIBED
523*                IN ROW I OF ARRAY N.  COLUMN ONE OF P CONTAINS THE
524*                NUMBER OF INPUTS TO THE ELEMENT WHILE COLUMN TWO
525*                CONTAINS THE POINTER TO THE ELEMENT INPUT VARIABLE
526*                LIST STORED IN ARRAY 1.
527*
540*        P1      A TEMPORARY VARIABLE USED TO REPRESENT THE INPUT
541*                LIST LENGTH OF AN ELEMENT DURING INPUT OF FAULTS.
542*
550*        P2      A TEMPORARY VARIABLE USED TO REPRESENT THE POINTER
551*                TO AN ELEMENT INPUT VARIABLE LIST DURING FAULT INPUT.
552*
560*        Q       A TWO DIMENSIONAL IMPLICIT STRING ARRAY USED TO
561*                STORE PAIRS OF ELEMENT NAMES THAT ARE CROSS
562*                COUPLED IN THE CIRCUIT.  EACH ROW CORRESPONDS TO
563*                A CROSS-COUPLED PAIR.  THE NAME OF ONE ELEMENT OF
564*                THE PAIR IS STORED IN COLUMN ONE AND THE NAME OF
565*                THE OTHER ELEMENT IS STORED IN COLUMN TWO.
566*
570*        Q1      A POINTER TO THE Q ARRAY.  DURING CIRCUIT ENTRY, Q1
571*                REPRESENTS THE NEXT ROW OF Q TO BE LOADED.  THE FINAL
572*                VALUE OF Q1 IS THE NUMBER OF ROWS USED IN Q.
573*
580*        R9      A SWITCH INDICATING WHETHER OR NOT RACE ANALYSIS
581*                WAS SELECTED.  R9=0: WAS SELECTED(DEFAULT VALUE).
582*                R9=1: NOT SELECTED.
583*
590*        S       A SWITCH INDICATING WHICH CHANNEL TO USE WHEN
591*                ACCESSING FUNCTION FILES.
592*
600*      V1,V2     TEMPORARY VARIABLES USED TO REPRESENT THE NUMERIC
601*                EQUIVALENTS OF CROSS-COUPLED ELEMENTS.
602*
610*        V3      A TEMPORARY VARIABLE USED TO COUNT THE NUMBER OF
611*                CHARACTERS IN A SUBSTRING.
612*
620*        W       A TEMPORARY VARIABLE USED AS A SWITCH INDICATING
621*                WHETHER OR NOT A MATCH WAS FOUND DURING STARTING
622*                STATE ENTRY.
623*
630*        X       USED AS A POINTER TO CHARACTERS IN Z$ DURING THE
631*                PARSING OF Z$.  IN ANOTHER CASE, IS USED TO
632*                INDICATE WHICH FUNCTION FILES ARE TO BE LOADED.
633*                X=1: LOAD ONE-FNS.  X=2: LOAD ZERO-FNS.
634*
640*        X$      A TEMPORARY VARIABLE USED STORE A FUNCTION NAME.
641*                IN ANOTHER APPLICATION, USED TO STORE 0 OR 1
642*                FUNCTION VALUES.
643*
650*        X1      USED AS AN INDEX VARIABLE IN FOR-NEXT LOOPS.
```

```
651*              IN ANOTHER APPLICATION, USED AS A POINTER TO
652*              A FUNCTION RECORD.
653*
660*        Y     USED AS AN INDEX VARIABLE IN FOR-NEXT LOOPS.
661*
670*        Y5    A PARAMETER REPRESENTING THE MAXIMUM NUMBER OF
671*              TERMS THAT CAN BE STORED IN THE FUNCTION FILE OF
672*              EACH FUNCTION.
673*
680*        Y$    A TEMPORARY VARIABLE USED TO REPRESENT PRODUCT TERMS
681*              DURING THE PARSING OF A USER ENTERED STARTING STATE
682*              FUNCTION.  IN ANOTHER APPLICATION, USED TO STORE
683*              0 OR 1 FUNCTION VALUES.
684*
690*        Z     USED AS A COUNTER OF THE NUMBER OF INPUTS OF AN
691*              ELEMENT DURING CIRCUIT ENTRY.
692*
700*        Z$    A TEMPORARY VARIABLE USED AS A STRING INPUT VAR.
800*****************************************************************
801*
802*         SUBROUTINE DESCRIPTIONS
803*
804*****************************************************************
805*
809*          PROGRAMMER DEFINED SUBROUTINES
810*
811*     LINE          DESCRIPTION
812*
813*
814*     2000      INPUTS CIRCUIT DESCRIPTION FROM KEYBOARD
815*
816*     3000      INPUTS FAULT DESCRIPTION FROM KEYBOARD
817*
818*     4000      INPUTS STARTING STATE CONDITIONS FROM KB
819*
820*     6000      INPUTS CIRCUIT DESCRIPTION FROM FILE
821*
900*****************************************************************
901*
902*          DIMENSION STATEMENTS
903*
904*****************************************************************
905*
910       DIM N(200,4), I(400,3), G(50,2), P(200,2)
1000****************************************************************
1001****************************************************************
1002*                                                              *
1003*          START OF MAIN PROGRAM                               *
1004*                                                              *
1005****************************************************************
1006****************************************************************
1007*
```

```
1008*
1050 OPEN "OLD1" TO :1, INPUT UPDATE\*OPEN FILE FOR OLD ONE-FNS.
1060 OPEN "OLD0" TO :2, INPUT UPDATE\*OPEN FILE FOR OLD ZERO-FNS.
1070 OPEN "NET" TO :3, INPUT UPDATE \*OPEN FILE FOR NET DESCRIPTION.
1100 H9=400\              ***SET LOOP LIMIT PARAMETER.
1105 Y5=100\              ***SET LIMIT ON # OF TERMS IN FUNCTIONS.
1120 S=0\                 ***INITIALIZE FUNCTION FILE SWITCH.
1125 INPUT=$\             ***SET SEQUENTIAL FILE ACCESS MODE.
1129*****PRINT SIMULATION MODE SELECTION HEADER. ***************.
1130 PRINT"SELECT SIMULATION MODE.  <CR>=1."
1135 PRINT"     (1) NEW CIRCUIT"
1140 PRINT"     (2) NEW FAULT"
1145 PRINT"     (3) NEW STARTING STATE"
1150 PRINT"     (4) CIRCUIT DESCRIPTION ON FILE"
1155 INPUT M0\            ***ENTER MODE NUMBER FROM KB.
1160 IF M0<=4 THEN 1175\     ***CHECK FOR VALID MODE NUMBER.
1164*****PRINT ERROR-MESSAGE IF INVALID MODE NUMBER ENTERED.
1165 PRINT"INVALID CHOICE.  MAKE ANOTHER SELECTION."
1170 ***REPEAT FOR NEW ENTRY.\   GOTO 1155
1174*****PRINT OUTPUT MODE SELECTION HEADER. *****
1175 PRINT"SELECT OUTPUT MODE.  <CR>=1."
1180 PRINT"     (1) PRINT STABLE EQUATION SETS ONLY"
1185 PRINT"     (2) PRINT ALL EQUATION SETS"
1190 INPUT M1\            ***ENTER OUTPUT MODE NUMBER.
1195 IF M1<=2 THEN 1210\**CHECK FOR VALID MODE NUMBER.
1199*****PRINT ERROR-MESSAGE IF INVALID NUMBER ENTERED.*****
1200 PRINT"INVALID CHOICE.  MAKE ANOTHER SELECTION."
1205 ***REPEAT FOR NEXT ENTRY.\GOTO 1190
1209*****PRINT RACE ANALYSIS SELECTION HEADER.*****
1210 PRINT"RACE ANALYSIS(YES OR NO).  <CR>=YES.";
1215 INPUT Z$\            ***ENTER RACE-ANALYSIS SELECTION.
1220 IF Z$="YES" THEN 1240\*CHECK FOR YES ENTRY.
1225 IF Z$="" THEN 1240\     ***CHECK FOR DEFAULT YES.
1230 R9=1\                ***SET SWITCH FOR NO RACE ANALYSIS.
1235 ***SKIP ALTERNATIVE SWITCH SETTING.\GOTO 1245
1240 R9=0\                ***SET SWITCH FOR RACE ANALYSIS.
1244*****BRANCH TO PROPER INITIACIZATION ROUTINE.*****
1245 ON M0 GOTO 1250,1350,1355,1250
1250 MAT P= ZER\          ***CLEAR THE INPUT POINTER ARRAY.
1255 FOR X=1 TO H9/2\     ***CLEAR THE ELEMENT ARRAY.
1260       FOR Y=1 TO 4
1265          N(X,Y)=""
1270       NEXT Y
1275 NEXT X
1280 FOR X=1 TO H9\       ***CLEAR THE INPUT LIST ARRAY.
1285       FOR Y=1 TO 3
1290          I(X,Y)=""
1295       NEXT Y
1300 NEXT X
1305 FOR X=1 TO H9/8\     ***CLEAR XC ELEMENT ARRAY.
1310       FOR Y=1 TO 2
1315          Q(X,Y)=""
```

```
1320    NEXT Y
1325 NEXT X
1330 IF M0<>4 THEN 1345\***CHECK FOR MODE 4***
1335***IF MODE 4 GET CIRCUIT FROM FILE***\GOSUB 6000
1340***SKIP KB ENTRY SUBR***\GOTO 1350
1345***CALL KB ENTRY SUBR***\GOSUB 2000
1350***CALL FAULT ENTRY SUBR\GOSUB 3000
1355***CALL CIRCUIT INITIALIZATION SUBR***\GOSUB 4000
1360 FOR X=1 TO I0\***PRINT CONTENTS OF I ARRAY.
1365     PRINT I(X,1),I(X,2),I(X,3)
1370 NEXT X
1375 CLOSE :1\***CLOSE OLD ONE FUNCTION FILE.***
1380 CLOSE :2\***CLOSE OLD ZERO FUNCTION FILE***
1385 CLOSE :3\***CLOSE CIRCUIT DESC FILE***
1390 C0=1
1395 ***TRANSFER TO PROGRAM SIM2.\CHAIN LINK "SIM2"
2000**********************************************************************
2001**********************************************************************
2002*          SUBROUTINES                                               *
2003**********************************************************************
2004**********************************************************************
2005*  SUBROUTINE TO ENTER CIRCUIT DESCRIPTION FROM KEYBOARD              *
2006**********************************************************************
2008 I1=1\              ***INITIALIZE FILE RECORD POINTER***
2009***PRINT ELEMENT ENTRY HEADER MESSAGE**********************
2010 PRINT"ENTER--ELEMENT NAME, TYPE, AND INPUT LIST."
2013 PRINT"SEPARATE ENTRIES WITH A SPACE."
2014 PRINT"ENTER<CR>TO TERMINATE ENTRY."
2015 I0=1\              ***INITIALIZE I ARRAY POINTER.
2017 N1=0\              ***INITIALIZE POINTER TO N AND P ARRAYS.
2020 Z$=""\             ***CLEAR Z$.
2022 INPUT Z$\          ***ENTER ELEMENT DESCRIPTION FROM KB.
2023 PRINT :3;I1,Z$\    ***PLACE ELEMENT DESCRIPTION IN FILE.
2024 I1=I1+1\           ***INC FILE RECORD POINTER.
2026 IF Z$=" " THEN 2030\*LOOK FOR END OF ENTRIES.
2030 N1=N1+1\           ***INC N ARRAY POINTER.
2034***** BEGIN PARSING OF Z$ ***************************************
2035 B$=""\             ***CLEAR B$.***
2040 X=1\               ***INITIALIZE POINTER TO Z$.***
2050 IF Z$(:X,1)=" " THEN 2090\*END OF ELEMENT NAME FIELD?***
2060 B$=B$+Z$(:X,1)\    ***IF END NOT REACHED, BUILD NAME IN B$.***
2070 X=X+1\             ***INC POINTER TO Z$ .
2080 ***REPEAT FOR NEXT CHARACTER.\GOTO 2050
2090 N(N1,1)=B$\        ***PLACE NAME IN ARRAY WHEN COMPLETE.
2100 N(N1,4)="NO"\      ***LABEL ELEMENT AS NO PRIMARY OUTPUT.
2110 B$=""\             ***CLEAR B$ FOR NEXT FIELD.
2120 X=X+1\             ***INC POINTER TO Z$.
2130 IF Z$(:X,1)=" " THEN 2170\*END OF ELEMENT-TYPE FIELD?
2140 B$=B$+Z$(:X,1)\    ***IF END NOT REACHED, BUILD NAME IN B$.
2150 X=X+1\             ***INC POINTER TO Z$.
2160 ***REPEAT FOR NEXT CHARACTER IN Z$.\GOTO 2130
2170 N(N1,2)=B$\        ***PLACE TYPE IN ARRAY WHEN COMPLETE.
```

```
2171 IF B$<>"OFF" THEN 2176\*CHECK FOR OFF ELEMENT ENTRY.
2172 PRINT"OFF SIMULATION ROUTINE NOT YET IMPLEMENTED."
2173 STOP
2176 Z=1                  \*INITIALIZE COUNT OF ELEMENT INPUTS.
2177 P(N1,2)=I0\          ***PLACE INPUT LIST POINTER IN P ARRAY.
2180 X=X+1\               ***INC POINTER TO Z$.
2190 B$=""\               ***CLEAR B$ FOR NEXT FIELD.
2200 IF Z$(:X,1)=" " THEN 2250\*END OF ELEMENT-INPUT SUBFIELD?
2210 B$=B$+Z$(:X,1)\      ***END NOT REACHED, BUILD NAME.
2230 X=X+1\               ***INC POINTER TO Z$.
2240 ***REPEAT FOR NEXT CHARACTER IN Z$.\GOTO 2200
2250 I(I0,1)=B$\          ***INPUT NAME COMPLETE. PLACE IN ARRAY I.
2260 Z=Z+1\               ***INC INPUT COUNT.
2270 I0=I0+1\             ***INC I ARRAY POINTER.
2280 L=LEN(Z$)\           ***COMPUTE LENGTH OF Z$.
2285 IF X<L THEN 2180\    ***LOOK FOR END OF Z$.
2290 P(N1,1)=Z-1\         ***END REACHED.  LOAD INPUT COUNT IN P.
2300 ***REPEAT FOR NEXT ELEMENT ENTRY.\GOTO 2020.
2430 FOR X=1 TO I0-1\     ***ASSIGN NUMERIC CODE TO ALL
2431     FOR Y=1 TO N1\   ***ELEMENT INPUTS EXCEPT PRIMARY INPUTS.
2432         IF I(X,1)<>N(Y,1) THEN 2436
2433         Z$=STR(Y)
2434         I(X,3)=Z$
2435         GOTO 2437
2436     NEXT Y
2437 NEXT X
2439***** PRINT PRIMARY INPUT NAME ENTRY HEADER. ****************
2440 PRINT"ENTER PRIMARY INPUT LINE NAMES.  TERMINATE WITH<CR>."
2442 I0=I0-1\             ***COMPUTE NUMBER OF ROWS IN I ARRAY.
2445 INPUT Z$\            ***ENTER PRIMARY INPUT NAME FROM KB.
2446 PRINT :3;I1,Z$\     ***PLACE INPUT NAME IN FILE.
2447 I1=I1+1\            ***INC FILE RECORD POINTER.
2449 IF Z$=" " THEN 2465\*LOOK FOR END OF INPUT NAME ENTRY.
2450 FOR X=1 TO I0\      ***SCAN I ARRAY FOR PRIMARY INPUT NAMES.
2455     IF Z$<>I(X,1) THEN 2460
2457     I(X,3)="YES"\   ***FLAG WITH YES WHEN FOUND.
2460 NEXT X
2462 ***REPEAT FOR NEXT INPUT-NAME ENTRY.\GOTO 2445
2464***** PRINT PRIMARY-OUTPUT NAME ENTRY HEADER. ****************
2465 PRINT"ENTER PRIMARY OUTPUT LINE NAMES.  TERMINATE WITH<CR>."
2467 INPUT Z$\            ***ENTER PRIMARY-OUTPUT NAME FROM KB.
2468 PRINT :3;I1,Z$\      ***PLACE NAME IN FILE.
2469 I1=I1+1\            ***INC FILE RECORD POINTER.
2471 IF Z$=" " THEN 2500\*LOOK FOR END OF OUTPUT NAME ENTRIES.
2472 FOR X=1 TO N1\       ***SCAN N ARRAY FOR PRIMARY OUTPUT NAMES.
2475     IF Z$<>N(X,1) THEN 2480
2477     N(X,4)="YES"\    ***FLAG WITH "YES" WHEN FOUND.
2480 NEXT X
2485*****REPEAT FOR NEXT OUTPUT NAME ENTRY.\GOTO 2467
2499***** PRINT CROSS-COUPLED ELEMENTS ENTRY HEADER. **************
2500 PRINT"ENTER--CROSS-COUPLED ELEMENT PAIRS.  SEPARATE NAMES"
2501 PRINT"WITH A SPACE.  ENTER<CR> TO TERMINATE."
```

```
2505 Q1=0\              ***INITIALIZE POINTER TO Q ARRAY.
2510 INPUT Z$\          ***ENTER CROSS-COUPLED NAMES FROM KB.
2512 PRINT :3;I1,Z$\    ***PLACE NAME PAIR IN FILE.
2514 I1=I1+1\           ***INC POINTER TO FILE RECORD.
2521 IF Z$=" " THEN 2700\*LOOK FOR END OF ENTRIES.
2530 Q1=Q1+1\           ***INC Q ARRAY POINTER.
2540 L=LEN(Z$)\         ***COMPUTE LENGTH OF Z$.
2550 FOR X=1 TO L\      ***BEGIN PARSE OF Z$.
2560    IF Z$(:X,1)=" " THEN 2580\*LOOK FOR END OF FIRST NAME.
2570 NEXT X
2580 B$=Z$(:1,X-1)\     ***LET B$= FIRST NAME.
2590 C$=Z$(:X+1)\       ***LET C$=SECOND NAME.
2640 FOR X1=1 TO N1\    ***SCAN N ARRAY FOR NAME IN PAIR.
2641    IF B$=N(X1,1) THEN 2645
2643 NEXT X1
2645 Z$=STR(X1)\        ***CONVERT ROW NUMBER WHERE FOUND TO STRING.
2646 Q(Q1,1)=Z$\        ***LOAD STRING IN Q ARRAY.
2680 FOR X1=1 TO N1\    ***SCAN N ARRAY FOR SECOND NAME IN PAIR.
2681    IF C$=N(X1,1) THEN 2685
2682 NEXT X1
2685 Z$=STR(X1)\        ***CONVERT ROW NUMBER WHERE FOUND TO STRING.
2686 Q(Q1,2)=Z$\        ***PLACE ROW NUMBER STRING IN Q ARRAY.
2690***** REPEAT FOR NEXT PAIR ENTRY.\GOTO 2510
2700 PRINT :3;I1,"."\   ***PLACE END OF DATA MARK IN FILE.
2710 RETURN\            ***RETURN TO MAIN PROGRAM(1350).
3000*************************************************************
3001*      SUBROUTINE TO INPUT FAULT DESCRIPTION FROM KB.       *
3002*************************************************************
3010 FOR X=1 TO N1\     **INITIALIZE ALL ELEMENT OUTPUTS AS FAULT FREE.
3020    N(X,3)="FF"
3030 NEXT X
3040 FOR X=1 TO I0\     ***INITIALIZE ALL INPUTS AS FAULT FREE.
3050    I(X,2)="FF"
3060 NEXT X
3069***** PRINT FAULT SIMULATION MODE SELECTION HEADER.
3070 PRINT"FAULT SIMULATION(YES/NO).  <CR>=NO.";
3072 INPUT Z$\          ***ENTER SELECTION.
3074 IF Z$="YES" THEN 3076\*LOOK FOR YES ENTRY.
3075***IF FAULT SIMULATION NO SELECTED, RETURN.\GOTO 3500
3076 PRINT"SELECT FAULT CLASS OR ENTER <CR> TO TERMINATE."
3077*****PRINT FAULT-CLASS SELECTION HEADER.
3080 PRINT"     (1) OUTPUT FAULT"
3081 PRINT"     (2) PRIMARY INPUT FAULT"
3082 PRINT"     (3) ELEMENT INPUT FAULT"
3083 PRINT"FAULT CLASS";
3084 INPUT Z$\          ***ENTER FAULT-CLASS SELECTION.
3085*****TRANSFER TO ROUTINE FOR CLASS SELECTED.
3086 IF Z$=" " THEN 3500\*LOOK FOR TERMINATION OF ENTRIES.
3087 IF Z$="1" THEN 3100
3088 IF Z$="2" THEN 3200
3089 IF Z$="3" THEN 3340
3090 PRINT"INVALID CHOICE.  MAKE ANOTHER SELECTION.";
```

```
3091***** PRINT ERROR-MESSAGE IF INVALID MODE NUMBER ENTERED. *******
3094*****RETURN FOR REPEAT ENTRY.\GOTO 3084
3099*****PRINT HEADER REQUESTING OUTPUT NAME ENTRY. **********
3100 PRINT"ENTER LINE NAME OR <CR> TO TERMINATE ENTRIES.";
3110 PRINT"LINE NAME";
3115 INPUT Z$\          ***ENTER LINE NAME.
3121 IF Z$=" " THEN 3083\*LOOK FOR END OF ENTRIES.
3124***** PRINT FAULT CONDITION ENTRY HEADER. ******
3125 PRINT"ENTER FAULT CONDITIONS";
3130 PRINT"S0/S1";
3135 INPUT B$\          ***ENTER FAULT-CONDITION.
3140 FOR X=1 TO N1\     ***SCAN N ARRAY FOR FAULTED LINE.
3150    IF Z$<>N(X,1) THEN 3170
3160    N(X,3)=B$\      ***FLAG ELEMENT WITH FAULT CONDITION.
3170 NEXT X
3180***REPEAT FOR ENTRY OF NEXT LINE-NAME TO BE FAULTED.\GOTO 3110
3199*** PRINT HEADER REQUESTING ENTRY OF PRIMARY INPUT NAME.*********
3200 PRINT"ENTER LINE NAME OR <CR> TO TERMINATE.";
3220 PRINT"LINE NAME";
3225 INPUT Z$\          ***ENTER PRIMARY INPUT NAME.
3241 IF Z$=" " THEN 3083\*LOOK FOR END OF ENTRIES.
3249***PRINT HEADER REQUESTING ENTRY OF FAULT-CONDITIONS.
3250 PRINT"ENTER FAULT CONDITIONS";
3260 PRINT"S0/S1";
3265 INPUT B$\          ***ENTER FAULT-CONDITION.
3270 FOR X=1 TO I0\     ***SCAN I ARRAY FOR LINE TO BE FAULTED.
3280    IF Z$<>I(X,1) THEN 3300
3290    I(X,2)=B$\      ***FLAG WITH FAULT-CONDITION WHEN FOUND.
3300 NEXT X
3305 GOTO 3220
3339*** PRINT HEADER REQUESTING ELEMENT INPUT NAME TO BE FAULTED.****
3340 PRINT"ENTER ELEMENT INPUT NAME OR <CR> TO TERMINATE.";
3350 PRINT"INPUT NAME";
3351 INPUT Z$\          ***ENTER INPUT LINE NAME.
3356 IF Z$=" " THEN 3083\*REPEAT FOR NEW FAULT CLASS.
3359*PRINT HEADER REQUESTING OUTPUT NAME OF ELEMENT WITH FAULTED INPUT.
3360 PRINT"ELEMENT ";
3370 PRINT"OUTPUT NAME";
3375 INPUT B$\          ***ENTER ELEMENT OUTPUT NAME.
3379***PRINT MESSAGE REQUESTING FAULT CONDITION ENTRY.**********
3380 PRINT"FAULT CONDITIONS";
3385 PRINT"S0/S1";
3390 INPUT C$\          ***ENTER FAULT CONDITION.
3400 FOR X=1 TO N1\     ***SCAN N ARRAY FOR INVOLVED ELEMENT.
3410    IF B$=N(X,1) THEN 3430
3420 NEXT X
3430 X0=X\              ***SAVE ROW NUMBER WHERE ELEMENT FOUND.
3432 P1=P(X,1)\         ***GET INPUT LIST LENGTH FOR ELEMENT.
3434 P2=P(X,2)\         ***GET POINTER TO INPUT LIST.
3440 FOR X=P2 TO P2+P1-1\*SCAN INPUT LIST FOR FAULTED INPUT.
3450    IF Z$<>I(X,1) THEN 3470
3460    I(X,2)=C$\      ***FLAG WITH FAULT CONDITION WHEN FOUND.
```

```
3470 NEXT X
3480***REPEAT FOR NEXT INPUT NAME TO BE FAULTED.\GOTO 3350
3500 RETURN\              ***RETURN TO MAIN PROGRAM (1355).
4000*************************************************************************
4001*    SUBROUTINE TO INPUT STARTING STATE CONDITIONS.                      *
4002*************************************************************************
4003*
4009*** PRINT STARTING STATE MODE SELECTION HEADER. ***
4010 PRINT"ENTER STARTING STATE MODE SELECTION.  <CR>=1."
4012 PRINT""
4014 PRINT"     (1) ALL UNKNOWNS"
4016 PRINT"     (2) DOUBLE CROSS-COUPLED GATE VARIABLES"
4018 PRINT"     (3) ALL VARIABLES"
4020 PRINT"     (4) USER SPECIFIED"
4022 PRINT"     (5) SINGLE CROSS-COUPLED GATE VARIABLE"
4024 PRINT"     (6) SPECIFIED CONSTANTS"
4026 PRINT"     (7) FIXED INPUT VALUES"
4028 PRINT"     (8) MULTIMODE"
4030 INPUT A              \**ENTER MODE SELECTION NUMBER.
4040 IF A<=8 THEN 4050 \**CHECK FOR VALID MODE NUMBER.
4044 ***PRINT ERROR=MESSAGE IF INVALID NUMBER ENTRED.***
4045 PRINT"INVALID CHOICE.  MAKE ANOTHER SELECTION."
4047 ***RETRY MODE ENTRY.\GOTO 4030
4050 FOR X=1 TO 2         \**IF VALID MODE SELECTED, INITIALLY SPECIFY
4051                      ***ALL ELEMENT OUTPUTS AS UNKNOWN.
4060    FOR Y=1 TO N1     \**SCAN ELEMENT NAMES IN N ARRAY.
4080       IF X=1 THEN 4110\*IF X=1 SPECIFY ONE FUNCTIONS.
4081                         **OTHERWISE, SPECIFY ZERO FUNCTIONS.
4090       X$=N(Y,1)+"-0="          \*CONSTRUCT ZERO FUNCTION NAME.
4100       *LOAD FUNCTION IN FILE.\GOTO 4120
4110       X$=N(Y,1)+"0="           \*CONSTRUCT ONE FUNCTION NAME.
4120       PRINT :X;YS*(Y-1)+1,X$\*LOAD FN NAME IN FIRST RECORD.
4125       PRINT :X,"0"             \*LOAD FN VALUE IN SECOND RECORD.
4126       PRINT :X,"."             \*LOAD END-OF-FN MARK IN 3RD RECORD.
4130    NEXT Y
4140 NEXT X
4149 ***TRANSFER TO APPROPRIATE ROUTINE FOR MODE NUMBER ENTERED.
4150 ON A GOTO 4950, 4250,4500,4750,4250,4900,5000,4200
4151 ***TRANSFER FOR DEFAULT MODE.\GOTO 4950
4199 ***PRINT HEADER FOR MULTIMODE SELECTION.
4200 PRINT"SELECT MODE (2),(3),(4),(5),OR (6)."
4202 PRINT"MODE (7) IS THEN SELECTED AUTOMATICALLY."
4210 INPUT A1             \**ENTER MODE NUMBER FOR MULTIMODE.
4219 ***TRANSFER TO PROPER ROUTINE FOR MODE SELECTED.
4220 ON A1-1 GOTO 4250,4500,4750,4250,4900
4250 FOR X=1 TO 2         \**BEGIN ROUTINE TO LOAD DOUBLE OR SINGLE
4251                      ***CROSS-COUPLED(XC) ELEMENT VARIABLES.
4270    FOR Y=1 TO N1     \**SCAN ELEMENTS IN N ARRAY.
4275       W=0            \**INITIALIZE MATCH FLAG WITH NO-MATCH VALUE.
4280       FOR Z=1 TO Q1\*SCAN ELEMENTS IN Q ARRAY.
4282          V1=VAL(Q(Z,1))\*GET XC ELEMENT NUMBER.
4284          V2=VAL(N(Z,2))\*GET XC ELEMENT NUMBER.
```

```
4290          IF V1=Y THEN 4300\*LOOK FOR MATCH WITH ELEMENT
4293          IF V2=Y THEN 4300\*IN N ARRAY.
4294          *IF NO MATCH, CONTINUE SCAN THROUGH Q ARRAY.\GOTO 4350
4300          IF X=1 THEN 4330\*IF X=1,LOAD ONE FNS. ELSE LOAD ZERO FNS.
4310          IF A=5 THEN 4312\*CHECK FOR SINGLE VARIABLE MODE.
4311          IF A1<>5 THEN 4313\*LOOK FOR SINGLE VAR MODE IN MULTIMODE.
4312          IF Y=V2 THEN 4316 \*CHECK FOR 2ND ELEMENT IN PAIR.
4313          X$=N(Y,1)+"-0"     \*CONSTRUCT OFF VAR FOR 1ST ELEMENT.
4314          *SKIP NEXT CONSTRUCTION.\GOTO 4318
4316          X$=N(V1,1)+"0"     \*CONSTRUCT ON VAR FOR 2ND ELEMENT.
4318          W=1                \*SET MATCH FLAG TO MATCH VALUE.
4320          *EXIT LOOP SINCE MATCH FOUND.\GOTO 4360
4330          IF A=5 THEN 4332   \*CHECK FOR SINGLE VARIABLE MODE.
4331          IF A1<>5 THEN 4333\*LOOK FOR SINGLE VAR MODE IN MULTIMODE.
4332          IF Y=V2 THEN 4336 \*CHECK FOR SECOND ELEMENT IN PAIR.
4333          X$=N(Y,1)+"0"      \*CONSTRUCT ON VAR FOR 1ST ELEMENT.
4334          *SKIP NEXT CONSTRUCTION.\GOTO 4338
4336          X$=N(V1,1)+"-0"    \*CONSTRUCT OFF VAR FOR 2ND ELEMENT.
4338          W=1                \*SET MATCH FLAG TO MATCH VALUE.
4340          *EXIT LOOP SINCE MATCH FOUND.\GOTO 4360
4350       NEXT Z
4360       IF W=0 THEN 4370\*LOOK FOR NOMATCH. SKIP LOAD IF NOMATCH.
4365       PRINT :X;Y5*(Y-1)+2,X$\*LOAD VAR IN 2ND RECORD OF FN.
4370    NEXT Y
4380 NEXT X
4385 IF A=8 THEN 5000  \***IF MULTIMODE, XFR TO FIXED INPUT ROUTINE.
4390 RETURN            \***RETURN TO MAIN PROGRAM (1360)
4500 FOR X=1 TO 2      \*BEGIN ROUTINE TO LOAD VARIABLES FOR ALL
4501                   **ELEMENT OUTPUTS.
4520    FOR Y=1 TO N1  \*SCAN N ARRAY.
4530          IF X=1 THEN 4560\*IF X=1, LOAD ONE FUNCTIONS;
4531                          **  ELSE, LOAD ZERO FUNCTIONS.
4540          X$=N(Y,1)+"-0"   \*CONSTRUCT OFF VARIABLE.
4550          *SKIP CONSTRUCTION OF ON VARIABLE\GOTO 4570
4560          X$=N(Y,1)+"0"    \*CONSTRUCT ON VARIABLE.
4570          PRINT :X;Y5*(Y-1)+2,X$\*LOAD VAR IN 2ND RECORD OF FN.
4580    NEXT Y
4590 NEXT X
4595 IF A=8 THEN 5000  \*IF MULTIMODE, XFR TO FIXED INPUT ROUTINE.
4600 RETURN            \*RETURN TO MAIN PROGRAM (1360).
4749 *** BEGIN ROUTINE FOR LOADING USER SPECIFIED INITIAL CONDITIONS.
4750 PRINT"ENTER EQUATION OR ENTER <CR> TO TERMINATE ENTRY."
4751 X1=2              \*INITIALIZE FN RECORD POINTER.
4752 INPUT Z$          \*ENTER FN FROM KB.
4755 IF Z$="" THEN 4950\*LOOK FOR END OF ENTRIES.
4756 C$=""\X$=""\B$="" \*CLEAR TEMPORARY VARIABLES.
4761 V3=0              \*INITIALIZE COUNT OF CHARS IN FN NAME.
4762 FOR X=1 TO 100    \*COUNT CHARACTERS IN FN NAME.
4764    V3=V3+1        \*INC COUNTER.
4766    IF Z$(:X,1)="=" THEN 4768\*LOOK FOR END OF NAME.
4767 NEXT X
4768 FOR X=1 TO V3-1   \*EXTRACT FN NAME FROM EQN AND LOOK
```

```
4769                    **FOR ZERO FN ID(-).
4770    IF Z$(:X,1)="-" THEN 4780
4774    C$=C$+Z$(:X,1)
4776 NEXT X
4778 GOTO 4782
4780 X$="-"             \*SAVE SYMBOL FOR ZERO FN.
4782 V4=LEN(Z$)         \*COMPUTE LENGTH OF Z$.
4783 FOR X=V3+1 TO V4   \*EXTRACT FN VALUE FROM EQN.
4784    B$=B$+Z$(:X,1)
4786  NEXT X
4787 FOR Y=1 TO N1      \*SEARCH N ARRAY FOR FN NAME.
4788    IF C$=N(Y,1) THEN 4792
4789 NEXT Y
4790 PRINT"FN NAME DOES NOT MATCH ANY ELEMENT OUTPUT. PLEASE REENTER."
4791 ***REPEAT ENTRY.  \GOTO 4750
4792 IF X$="" THEN 4820\*CHECK FOR ONE FN SPECIFICATION.
4793 Y$=C$+"-0="        \*CONSTRUCT COMPLETE ZERO FUNCTION NAME.
4794 PRINT :2;Y5*(Y-1)+1,Y$\*LOAD FN NAME IN RECORD 1 OF FN.
4800 V4=LEN(B$)         \*COMPUTE LENGTH OF B$.
4801 Y$=""              \*CLEAR Y$ FOR PARSING OF B$.
4803 FOR X=1 TO V4      \*BEGIN PARSING OF B$ INTO TERMS.
4805    IF B$(:X,1)="+" THEN 4810\*END OF TERM?
4806    Y$=Y$+B$(:X,1) \*CONSTRUCT TERM UNTIL END REACHED.
4808    *IF END NOT REACHED, GET NEXT CHAR.\GOTO 4814
4810    PRINT :2;Y5*(Y-1)+X1,Y$\*LOAD TERM IN RECORD X1.
4812    Y$=""           \*CLEAR Y$ FOR NEXT TERM.
4813    X1=X1+1         \*INC RECORD POINTER.
4814 NEXT X
4815 PRINT :2;Y5*(Y-1)+X1,Y$\*LOAD LAST TERM.
4816 PRINT :2;Y5*(Y-1)+X1+1,"."\*LOAD END-OF-FN MARK.
4818 ***REPEAT FOR NEXT EQN ENTRY.\GOTO 4750
4820 Y$=C$+"0="         \*CONSTRUCT ONE-FN NAME.
4822 PRINT :1;Y5*(Y-1)+1,Y$\*LOAD FN NAME IN RECORD 1 OF FN.
4824 V4=LEN(B$)         \*COMPUTE LENGTH OF B$.
4826 Y$=""              \*CLEAR Y$ FOR PARSE OF B$.
4828 FOR X=1 TO V4      \*PARSE B$ INTO TERMS AND LOAD IN FILE.
4830    IF B$(:X,1)="+" THEN 4836\* END OF TERM?
4832    Y$=Y$+B$(:X,1) \*CONSTRUCT TERM, END NOT REACHED.
4834    *DO NOT LOAD UNTIL END REACHED.\GOTO 4840
4836    PRINT :1;Y5*(Y-1)+X1,Y$\*LOAD TERM IN RECORD X1.
4838    Y$=""           \*CLEAR Y$ FOR NEXT TERM.
4839    X1=X1+1         \*INC RECORD POINTER.
4840 NEXT X
4841 PRINT :1;Y5*(Y-1)+X1,Y$\*LOAD LAST TERM.
4842 PRINT :1;Y5*(Y-1)+X1+1,"."\*LOAD END-OF-FN MARK.
4844 *** REPEAT FOR ENTRY OF NEXT EQUATION.\GOTO 4750
4899 *** BEGIN ROUTINE FOR ENTRY OF SPECIFIED CONSTANTS.
4900 PRINT"ENTER--LINE NAME=0 OR 1--ENTER <CR> TO TERMINATE ENTRY."
4901 INPUT Z$            \*ENTER LINE NAME AND DESIRED VALUE.
4903 IF Z$=" " THEN 4950 \*END OF ENTRIES?
4904 V3=0                \*INITIALIZE COUNTER.
4905 FOR X=1 TO 100      \*COUNT NUMBER OF CHARS IN LINE NAME.
```

```
4906     V3=V3+1                \*INC COUNTER.
4907     IF Z$(:X,1)="=" THEN 4909\END OF NAME?
4908 NEXT X
4909 C$=Z$(:1,V3-1)             \*EXTRACT LINE NAME FROM Z$.
4910 B$=Z$(:V3+1)               \*EXTRACT LINE VALUE FROM Z$.
4911 FOR Y=1 TO N1              \*SCAN N ARRAY FOR LINE NAME.
4912     IF C$=N(Y,1) THEN 4916
4914 NEXT Y
4916 IF B$="1" THEN 4924\*TEST FOR VALUE OF ONE.
4918 X$="0"                     \*VALUE OF ZERO ENTERED.
4920 Y$="I"                     \*ASSIGN (0,1) TO FNS.
4922 *** LOAD FNS IN FILE.\ GOTO 4928
4924 X$="1"                     \*VALUE OF ONE ENTERED.
4926 Y$="0"                     \*ASSIGN (1,0) TO FNS.
4928 PRINT :1;Y5*(Y-1)+2,X$\*LOAD ONE-FN VALUE.
4930 PRINT :2;Y5*(Y-1)+2,Y$\*LOAD ZERO-FN VALUE.
4940 *** REPEAT FOR NEXT ENTRY.\ GOTO 4901
4950 IF A=8 THEN 5000   \*IF MULTIMODE, GO TO FIXED-INPUT ROUTINE.
4960 RETURN             \*RETURN TO MAIN PROGRAM (1360).
4999 *** BEGIN ROUTINE TO ENTER FIXED INPUT VALUES.
5000 FOR X=1 TO I0      \*ERASE PREVIOUSLY ENTERED VALUES.
5001     IF I(X,2)="S0" THEN 5004\*DO NOT REMOVE ANY
5002     IF I(X,2)="S1" THEN 5004\*SPECIFIED FAULTS.
5003     I(X,2)="FF"
5004 NEXT X
5005 IF A<8 THEN 5007   \*CHECK FOR MULTIMODE ENTRY.
5006 PRINT"ENTER FIXED INPUT VALUES."
5007 PRINT"ENTER LINE NAME =0 OR =1 OR TO STOP ENTER <CR>.";
5010 INPUT Z$           \*ENTER LINE NAME AND VALUE.
5020 IF Z$=" " THEN 5300\*END OF ENTRIES?
5030 V3=0               \*INITIALIZE COUNTER.
5040 FOR X=1 TO 100     \*COUNT THE CHARACTERS IN LINE NAME.
5050     V3=V3+1        \*INC COUNT.
5060     IF Z$(:X,1)="=" THEN 5080\*LOOK FOR END OF NAME.
5070 NEXT X
5080 C$=Z$(:1,V3-1)     \*EXTRACT LINE NAME FROM Z$.
5090 B$=Z$(:V3+1)       \*EXTRACT LINE VALUE FROM Z$.
5095 M=0                \*INITIALIZE FLAG FOR NO MATCH VALUE.
5100 FOR X=1 TO I0      \*SCAN I ARRAY FOR LINE NAME.
5110     IF C$<>I(X,1) THEN 5130
5120     I(X,2)=B$      \*INSERT VALUE WHEN MATCH FOUND.
5125     M=1            \*SET FLAG TO  MATCH VALUE.
5130 NEXT X
5132 IF M=1 THEN 5007   \*MATCH FOUND, REPEAT FOR NEXT ENTRY.
5133 PRINT"LINE NAME DOES NOT MATCH ANY PRIMARY INPUT OF CIRCUIT."
5134 PRINT"PLEASE TRY AGAIN."
5135 ***RETRY ENTRY.\ GOTO 5007
5300 RETURN             \* RETURN TO MAIN PROGRAM (1360).
6000*************************************************************
6001*    SUBROUTINE TO INPUT CIRCUIT DESCRIPTION FROM FILE.     *
6002*************************************************************
6005***THIS SUBR WAS DERIVED FROM AND IS SIMILAR TO SUBR 2000.***
```

```
6015 I0=1\                    ***INITIALIZE I ARRAY POINTER.
6017 N1=0\                    ***INITIALIZE N ARRAY POINTER.
6020 Z$=""\                   ***CLEAR Z$.
6022 INPUT :5,Z$\             ***INPUT ELEMENT DESCRIPTION FROM FILE.
6023 PRINT Z$\                ***PRINT ELEMENT DESCRIPTION.
6025 IF Z$=" " THEN 6450\*LOOK FOR END OF ELEMENT ENTRIES.
6030 N1=N1+1\                 ***INC N ARRAY POINTER.
6034***** BEGIN PARSING OF Z$ **********
6035 B$=""\                   ***CLEAR B$ FOR NEXT FIELD.
6040 X=1\                     ***INITIALIZE POINTER TO Z$.
6050 IF Z$(:X,1)=" " THEN 6090\*END OF ELEMENT-NAME FIELD?
6060 B$=B$+Z$(:X,1)\          ***END NOT REACHED.  ADD TO NAME.
6070 X=X+1\                   ***INC POINTER TO Z$.
6080 ****REPEAT FOR NEXT CHARACTER IN Z$.\GOTO 6050
6090 N(N1,1)=B$\              ***WHEN NAME COMPLETE, PLACE IN N ARRAY.
6100 N(N1,4)="NO"\            ***LABEL ALL ELEMENTS AS NO OUTPUT.
6110 B$=""\                   ***CLEAR B$ FOR NEXT FIELD.
6120 X=X+1\                   ***INC POINTER TO Z$.
6130 IF Z$(:X,1)=" " THEN 6170\*END OF ELEMENT TYPE FIELD?
6140 B$=B$+Z$(:X,1)\          ***END NOT REACHED, BUILD NAME.
6150 X=X+1\                   ***INC POINTER TO Z$.
6160***REPEAT FOR NEXT CHARACTER IN Z$.\GOTO 6130
6170 N(N1,2)=B$\              ***WHEN NAME COMPLETE, PLACE IN N ARRAY.
6175 Z=1\                     ***INITIALIZE COUNT OF ELEMENT INPUTS.
6177 P(N1,2)=I0\              ***LOAD POINTER TO INPUT LIST.
6180 X=X+1\                   ***INC POINTER TO Z$.
6190 B$=""\                   ***CLEAR B$ FOR NEXT FIELD.
6200 IF Z$(:X,1)=" " THEN 6250\*END OF INPUT NAME SUBFIELD?
6210 B$=B$+Z$(:X,1)\          ***END NOT REACHED.  RECONSTRUCT NAME.
6230 X=X+1\                   ***INC POINTER TO Z$.
6240 ***REPEAT FOR NEXT CHARACTER IN Z$.\GOTO 6200
6250 I(I0,1)=B$\              ***WHEN NAME COMPLETE, PLACE IN I ARRAY.
6260 Z=Z+1\                   ***INC COUNT OF ELEMENT INPUTS.
6270 I0=I0+1\                 ***INC POINTER TO I ARRAY.
6280 L=LEN(Z$)\               ***COMPUTE LENGTH OF Z$.
6285 IF X<L THEN 6180\****IF END NOT REACHED, GET NEXT INPUT.
6290 P(N1,1)=Z-1\             ***WHEN END REACHED, PLACE # OF INPUTS IN P.
6300 ***REPEAT FOR NEXT ELEMENT DESCRIPTION.\GOTO 6020
6430 FOR  X=1 TO I0-1\ ***ASSIGN NUMERIC CODE TO ALL INPUTS
6431     FOR Y=1 TO N1\ ***EXCEPT PRIMARY INPUTS.
6432          IF I(X,1)<>N(Y,1) THEN 6436
6433          Z$=STR(Y)
6434          I(X,3)=Z$
6435          GOTO 6437
6436     NEXT Y
6437 NEXT X
6442 I0=I0-1\                 ***COMPUTE THE NUMBER OF ROWS IN I.
6445 INPUT :5,Z$\             ***GET PRIMARY INPUT NAME FROM FILE.
6446 PRINT Z$\                ***PRINT PRIMARY INPUT NAME.
6448 IF Z$=" " THEN 6465\*LOOK FOR END OF PRIMARY INPUT ENTRIES.
6450 FOR X=1 TO I0\           ***SCAN I ARRAY FOR PRIMARY INPUT NAMES.
6455     IF Z$<>I(X,1) THEN 6460
```

```
6457     I(X,3)="YES"\   ***FLAG WITH YES WHEN FOUND.
6460 NEXT X
6462***REPEAT FOR NEXT INPUT NAME.\GOTO 6445.
6465 INPUT :3,Z$\          ***GET PRIMARY OUTPUT NAME FROM FILE.
6466 PRINT Z$\             ***PRINT PRIMARY OUTPUT NAME.
6470 IF Z$=" " THEN 6500\*LOOK FOR END OF OUTPUT NAME ENTRIES.
6472 FOR X=1 TO N1\        ***SCAN N ARRAY FOR PRIMARY OUTPUT NAMES.
6475     IF Z$<>N(X,1) THEN 6480
6477     N(X,4)="YES"\     ***FLAG WITH YES WHEN FOUND.
6480 NEXT X
6485 ***REPEAT FOR NEXT OUTPUT NAME.\GOTO 6465
6500 Q1=0
6510 INPUT :3,Z$\          ***GET XC ELEMENT PAIR FROM FILE.
6511 PRINT Z$\             ***PRINT ELEMENT PAIR.
6520 IF Z$=" " THEN 6700\*LOOK FOR END OF XC PAIR ENTRIES.
6530 Q1=Q1+1\              ***INC POINTER TO Q ARRAY.
6540 L=LEN(Z$)\            ***COMPUTE LENGTH OF Z$.
6550 FOR X=1 TO L\         ***BEGIN PARSING Z$.
6560     IF Z$(:X,1)=" " THEN 6580\*END OF FIRST NAME IN PAIR?
6570 NEXT X
6580 B$=Z$(:1,X-1)\        ***LET B$ = FIRST NAME.
6590 C$=Z$(:X+1)\          ***LET C$ = SECOND NAME.
6640 FOR X1=1 TO N1\       ***SCAN N ARRAY FOR NAMES OF XC ELEMENTS.
6641     IF B$=N(X1,1) THEN 6645
6643 NEXT X1
6645 Z$=STR(X1)\           ***GENERATE NUMERIC CODE FOR ELEMENT.
6646 Q(Q1,1)=Z$\           ***PLACE CODE IN Q ARRAY.
6680 FOR X1=1 TO N1\       ***SCAN N ARRAY FOR SECOND NAME IN XC PAIR.
6681     IF C$=N(X1,1) THEN 6685
6682 NEXT X1
6685 Z$=STR(X1)\           ***GENERATE NUMERIC CODE.
6686 Q(Q1,2)=Z$\           ***PLACE CODE IN Q ARRAY.
6690 ***REPEAT FOR NEXT PAIR OF XC ELEMENTS.\GOTO 6510
6700 RETURN\               ***RETURN TO MAIN PROGRAM (1340).
99999 END
```

10:10 OCT 11, '79 DC/SIM2DOC.BCARROLL

```
1*      SSSSS   III   M   M   LL      00000   GGGGG
2*      S        I    MM MM   L      0   0   G
3*      SSSSS    I    M M M   L      0   0   G  GG
4*          S    I    M   M   L      0   0   G   G
5*      SSSSS   III   M   M   LLLLL  00000   GGGGG
6*
7*           PROGRAM:        SIM2DOC
8*           VERSION:        SIGMA 5
9*           REVISION:       ORIGINAL
10*          DATE:           10/10/79
11*          PROGRAMMER:     B. D. CARROLL
12*
13*************************************************************************
14*
15*           CONTRACT SUPPORT
16*************************************************************************
17*
18*      THIS PROGRAM WAS DEVELOPED FOR NASA MARSHALL FLIGHT
19*      CENTER UNDER CONTRACT NAS8-31572.
20*
21*************************************************************************
22*
23*           MODIFICATION HISTORY
24*
25*************************************************************************
26*
39*
100************************************************************************
101*
102*           PROGRAM DESCRIPTION
103*
104************************************************************************
105*
106*     SIMLOG IS A PROGRAM FOR SIMULATING LOGIC CIRCUITS.
107*     BOTH COMBINATIONAL AND SEQUENTIAL CIRCUITS CAN BE
108*     SIMULATED.  CIRCUITS CAN BE SIMULATED FAULT-FREE OR
109*     WITH SINGLE OR MULTIPLE STUCK TYPE FAULTS.
110*     LOGIC ELEMENTS ACCOMMODATED BY THE SIMULATOR ARE
111*     NAND GATES AND NOR GATES.  OTHER ELEMENTS ARE TO
112*     BE ADDED.
113*
114*     TWO VERSIONS OF SIMLOG HAVE BEEN WRITTEN.
115*     ONE VERSION HAS BEEN WRITTEN IN BASIC-PLUS FOR
116*     EXECUTION ON A PDP11/40 RSTS/E SYSTEM.
117*     ANOTHER VERSION HAS BEEN WRITTEN IN XEROX BASIC
118*     FOR EXECUTION ON A SIGMA 5 CPV SYSTEM.
119*
120*     THE PDP11/40 VERSION IS PARTITIONED INTO TWO
121*     SUBPROGRAMS, SIMA AND SIMB.  THIS PARTITIONING WAS
122*     NECESSARY DUE TO THE LARGE MEMORY REQUIREMENTS OF
```

```
123*     THE PROGRAM.  VIRTUAL ARRAYS ARE USED FOR ALL LARGE
124*     ARRAY STORAGE.
125*
126*     THE SIGMA 5 VERSION IS PARTITIONED INTO THREE
127*     SUBPROGRAMS, SIM1, SIM2, AND RACE.  THIS DEGREE OF
128*     PARTITIONING WAS NECESSARY SINCE VIRTUAL ARRAYS ARE
129*     NOT AVAILABLE ON THE SIGMA 5.
130*
131*     THE FUNCTION OF SIM1 IS TO INPUT ALL DATA NEEDED
132*     CONCERNING CIRCUIT DESCRIPTION, FAULT DESCRIPTION,
133*     INITIAL CONDITIONS, AND SIMULATION MODES.
134*
135*     THE FUNCTION OF SIM2 IS TO PERFORM THE ACTUAL
136*     SIMULATION COMPUTATIONS WITH THE EXCEPTION OF
137*     RACE ANALYSIS AND TO OUTPUT THE SIMULATION RESULTS.
138*
139*     THE FUNCTION OF RACE IS TO PERFORM THE RACE
140*     ANALYSIS COMPUTATIONS.
141*
142*
300*****************************************************************************
301*
302*          I/O STREAMS
303*
304*****************************************************************************
305*
306*
309*     STREAM          USAGE
310*
311*
312*          1          FILE OF OLD ONE-FUNCTIONS(S=0)
313*                     FILE OF NEW ONE-FUNCTIONS(S=1)
314*
315*          2          FILE OF OLD ZERO-FUNCTIONS(S=0)
316*                     FILE OF NEW ZERO-FUNCTIONS(S=1)
317*
318*          3          FILE OF NEW ZERO-FUNCTIONS(S=0)
319*                     FILE OF OLD ZERO-FUNCTIONS(S=1)
320*
321*          4          FILE OF NEW ONE-FUNCTIONS(S=0)
322*                     FILE OF OLD ONE-FUNCTIONS(S=1)
323*
324*
325*          THE ABOVE REPRESENTS THE BASIC ASSIGNMENTS OF I/O STREAMS
326*     TO FUNCTION FILES.  HOWEVER, THE SIGMA 5 PROVIDES ONLY 4 I/O
327*     STREAM NUMBERS.  HENCE, IT BECOMES NECESSARY TO REASSIGN STREAMS
328*     WHEN ACCESS TO ARRAY-FILES IS NEEDED.  THIS REASSIGNMENT IS
329*     DYNAMIC AS A FUNCTION OF PARAMETERS S AND J SINCE IT IS
330*     NECESSARY TO KEEP ACCESS TO ONE FUNCTION FILE ALSO.
331*     THE FOLLOWING TABLE DETAILS THIS DYNAMIC ASSIGNMENT.
332*
333*-------!-------------------------!-----------------------------!
```

```
334*        !        S=0                    !              S=1                    !
335*--------!-------------------------------!-------------------------------------!
336*STREAM#! J=0        !     J=1           !  J=0         !     J=1              !
337*--------!------------!-------------------!--------------!---------------------!
338*   1    ! OLD1       ! FARRAY            ! FARRAY       ! GARRAY             !
339*--------!------------!-------------------!--------------!---------------------!
340*   2    ! FARRAY     ! OLD0              ! GARRAY       ! FARRAY             !
341*--------!------------!-------------------!--------------!---------------------!
342*   3    ! GARRAY     ! HARRAY            ! HARRAY       ! NEW0               !
343*--------!------------!-------------------!--------------!---------------------!
344*   4    ! HARRAY     ! GARRAY            ! NEW1         ! HARRAY             !
345*--------!------------!-------------------!--------------!---------------------!
346*
400*****************************************************************************
401*
402*          VARIABLE DEFINITIONS
403*
404*****************************************************************************
405*
409*     VARIABLE        DEFINITION
410*
411 *        A       USED AS AN INDEX FOR THE A$ ARRAY.
412 *
413 *        A$      A STRING ARRAY USED TO STORE THE LITERALS OF A
414 *                PRODUCT TERM THAT HAS BEEN PARSED.
415 *
416 *        B       USED AS THE CONTROL VARIABLE IN A FOR-NEXT LOOP.
417 *
418 *        B0      USED AS THE CONTROL VARIABLE IN A FOR-NEXT LOOP.
419 *
420 *        B1      THE CONTROL VARIABLE TERMINAL VALUE FOR A FOR-NEXT
421 *                LOOP IN A BUBBLE SORT PROCEDURE.
422 *
423 *         B$     USED AS A TEMPORARY STRING VARIABLE.
424 *
425 *         C      USED AS A FLAG, A COUNTER, AND AS A FOR-NEXT
426 *                LOOP CONTROL VARIABLE.
427 *
428 *         C0    USED AS A SWITCH THAT INDICATES(C0=2) WHEN ENTRY
429 *               TO SIM2 IS FROM RACE.
430 *
431 *         C$     USED AS A TEMPORARY STRING VARIABLE.
432 *
433 *         D      USED AS A FLAG TO INDICATE (D=1) THAT A MINUS
434 *                SIGN SHOULD BE INSERTED IN A TEST STRING.
435 *
436 *        D$      A STRING ARRAY (ONE-DIMENSIONAL) USED AS
437 *                TEMPORARY STORAGE DURING THE FUNCTION
438 *                MINIMIZATION ROUTINE.
439 *
440 *         F       INDICATES THE I/O STREAM TO WHICH F ARRAY FILE
441 *                 IS OPENED.
```

```
442 *
443 *          F5      USED AS A FLAG THAT INDICATES WHETHER OR NOT A
444 *                  MINUS (-) HAS BEEN ENCOUNTERED DURING THE PARSING
445 *                  OF A PRODUCT TERM.  F5=1 INDICATES THAT A MINUS
446 *                  HAS BEEN ENCOUNTERED.  LOCAL TO SUBR 7800.
447 *
448 *          F1      A FLAG USED TO INDICATE WHETHER AN INPUT FAULT (1)
449 *                  OR AN OUTPUT FAULT (2) IS TO BE INSERTED.
450 *
451 *          F2      WHEN AN INPUT FAULT IS TO BE INSERTED, F2=1
452 *                  INDICATES THAT THE FAULT IS ON THE FIRST INPUT OF
453 *                  AN ELEMENT.  F2=2 INDICATES ANY OTHER INPUT.
454 *
455 *          G       INDICATES THE I/O STREAM TO WHICH THE G ARRAY FILE
456 *                  IS OPENED.
457 *
458 *          G1      USED AS A TEMPORARY VARIABLE FOR F OR G.
459 *
460 *          H       INDICATES THE I/O STREAM TO WHICH THE H ARRAY FILE
461 *                  IS OPENED.
462 *
463 *          H9      INDICATES THE UPPER LIMIT OF THE LOOP VARIABLE
464 *                  OF SOME FOR-NEXT LOOPS.  SHOULD BE SET >= THE
465 *                  NUMBER OF ROWS IN THE I ARRAY.
466 *
467 *          I       A TWO DIMENSIONAL IMPLICIT STRING ARRAY USED TO
468 *                  STORE CIRCUIT ELEMENT INPUT LISTS.  EACH ROW OF
469 *                  THE ARRAY DESCRIBES ONE INPUT VARIABLE OF ONE
470 *                  ELEMENT OF THE CIRCUIT BEING DESCRIBED.  THE
471 *                  VARIABLE NAME IS GIVEN IN COLUMN ONE.  ANY FAULT
472 *                  CONDITION ASSOCIATED WITH THE INPUT IS
473 *                  GIVEN IN COLUMN TWO.  THE THIRD COLUMN INDICATES
474 *                  WHETHER OR NOT THE INPUT IS A PRIMARY INPUT.
475 *
476 *          I0      A POINTER TO THE NEXT ROW OF THE I ARRAY TO BE
477 *                  LOADED DURING INPUT OF THE CIRCUIT DESCRIPTION.
478 *                  THE FINAL VALUE OF I0 INDICATES THE NUMBER OF
479 *                  ROWS OF ARRAY I THAT ARE USED.
480 *
481 *          I1      THE CONTROL VARIABLE IN THE PRIMARY FOR-NEXT LOOP
482 *                  OF THE SIM2 MAIN PROGRAM.  IN THIS CONTEXT, IT
483 *                  SERVES AS A POINTER TO THE N AND P ARRAYS.
484 *
485 *          J       USED AS A SWITCH TO INDICATE WHICH FUNCTION TYPE IS
486 *                  BEING GENERATED.  J=0 INDICATES 0-FN.
487 *                  J=1 INDICATES 1-FN.
488 *
489 *          J$      USED AS A TEMPORARY STRING VARIABLE INDICATING THE
490 *                  ID NUMBER OF AN ELEMENT INPUT.
491 *
492 *          K       A SWITCH INDICATING WHICH ARRAY F (K=1) OR G (K=0)
493 *                  IS TO BE LOADED WITH A FUNCTION.
```

```
494 *
495 *        L        REPRESENTS THE LENGTH OF A STRING.  VALUE IS
496 *                 ESTABLISHED BY THE LEN FUNCTION.
497 *
498 *      L1,L2      USED TO INDICATE THE LENGTH OF TERMS DURING THE
499 *                 TERM ORDERING PROCEDURE.
500 *
501 *        N        A TWO DIMENSIONAL IMPLICIT STRING ARRAY USED TO
502 *                 STORE THE ELEMENT DESCRIPTIONS AND INTERCONNECTIONS
503 *                 OF A CIRCUIT.  EACH ROW OF THE ARRAY CORRESPONDS TO
504 *                 ONE ELEMENT OF THE CIRCUIT.  COLUMN ONE CONTAINS
505 *                 THE ELEMENT OUTPUT NAME.  COLUMN TWO IS THE ELEMENT
506 *                 TYPE.  COLUMN THREE INDICATES THE FAULT CONDITION
507 *                 OF THE ELEMENT OUTPUT.  COLUMN FOUR INDICATES
508 *                 WHETHER OR NOT THE ELEMENT OUTPUT IS A PRIMARY
509 *                 OUTPUT OF THE CIRCUIT.
510 *
511 *        N0       A FLAG INDICATING IF THE OLD FUNCTION SET AND THE
512 *                 NEW FUNCTION SET ARE EQUAL (N0=1) OR ARE NOT EQUAL
513 *                 (N0=0).
514 *
515 *        N1       INDICATES THE NUMBER OF ELEMENTS IN THE CIRCUIT.
516 *                 CONSEQUENTLY THE NUMBER OF ROWS IN N AND P.
517 *
518 *        O2       FLAG USED TO INDICATE IF SIMULATION IS TO BE
519 *                 CONTINUED FOLLOWING THE DETECTION OF A POSSIBLE
520 *                 OSCILLATION.  O2=1 MEANS TO CONTINUE WHILE
521 *                 O2=0 MEANS TO STOP.
522 *
523 *        P        A TWO DIMENSIONAL NUMERIC ARRAY USED TO STORE
524 *                 POINTERS TO THE INPUT LISTS STORED IN ARRAY I.
525 *                 ROW I OF P CORRESPONDS TO THE ELEMENT DESCRIBED
526 *                 IN ROW I OF ARRAY N.  COLUMN ONE OF P CONTAINS THE
527 *                 NUMBER OF INPUTS TO THE ELEMENT WHILE COLUMN TWO
528 *                 CONTAINS THE POINTER TO THE ELEMENT INPUT VARIABLE
529 *                 LIST STORED IN ARRAY I.
530 *
531 *        P0       USED AS AN INDEX TO THE INPUT LIST OF
532 *                 AN ELEMENT.  P0 IS ADDED TO THE INPUT LIST POINTER
533 *                 TO OBTAIN THE ROW OF I CORRESPONDING TO THE INPUT.
534 *
535 *        Q        A TWO DIMENSIONAL IMPLICIT STRING ARRAY USED TO
536 *                 STORE PAIRS OF ELEMENT NAMES THAT ARE CROSS
537 *                 COUPLED IN THE CIRCUIT.  EACH ROW CORRESPONDS TO
538 *                 A CROSS-COUPLED PAIR.  THE NAME OF ONE ELEMENT OF
539 *                 THE PAIR IS STORED IN COLUMN ONE AND THE NAME OF
540 *                 THE OTHER ELEMENT IS STORED IN COLUMN TWO.
541 *
542 *        Q1       THE NUMBER OF CROSS-COUPLED PAIRS OF ELEMENTS IN
543 *                 THE CIRCUIT.  ALSO THE NUMBER OF ROWS IN Q.
544 *
545 *        R        THE RIPPLE-TIME VARIABLE.
```

```
546 *
547 *        R0      RIPPLE-TIME LIMIT.  INPUT BY THE USER.
548 *
549 *        R1      THE VALUE OF RIPPLE-TIME (R) AT THE BEGINNING OF
550 *                A NEW INPUT-TIME (T).  NEEDED TO DETERMINE IF A
551 *                POSSIBLE OSCILLATION CONDITION EXISTS.
552 *
553 *        R9      A SWITCH INDICATING WHETHER OR NOT RACE ANALYSIS
554 *                WAS SELECTED.  R9=0: WAS SELECTED(DEFAULT VALUE).
555 *                R9=1: NOT SELECTED.
556 *
557 *        S       A SWITCH INDICATING WHICH CHANNEL TO USE WHEN
558 *                ACCESSING FUNCTION FILES.
559 *
560 *        T       INPUT-TIME VARIABLE.
561 *
562 *        T0      INPUT-TIME LIMIT.  INPUT BY USER.
563 *
564 *        U       A ONE-DIMENSIONAL, ONE-ELEMENT ARRAY USED IN THE
565 *                CONVERSION OF THE INPUT-TIME VALUE TO A STRING.
566 *
567 *        V       A TEMPORARY VARIABLE USED TO REPRESENT THE
568 *                NUMERIC VALUE OF A STRING VARIABLE.
569 *
570 *        W       USED AS A POINTER TO THE H (RESULT) ARRAY DURING
571 *                THE AND ROUTINE.
572 *
573 *        X       USED AS THE CONTROL VARIABLE IN FOR-NEXT LOOPS.
574 *
575 *        X0      A TEMPORARY VARIABLE USED TO STORE THE SORT LOOP
576 *                LIMIT IN THE TERM SORT ROUTINE.
577 *
578 *        X1      A TEMPORARY VARIABLE USED TO STORE THE NUMBER OF
579 *                TERMS IN THE FUNCTION BEING SORTED PLUS ONE.
580 *
581 *        X2
582 *
583 *        X3      USED AS THE CONTROL VARIABLE IN A FOR-NEXT LOOP.
584 *
585 *        X$      USED AS A TEMPORARY STRING VARIABLE TO STORE A
586 *                FAULT CONDITION TO BE INSERTED.  ALSO USED AS A
587 *                TEMPORARY VARIABLE TO STORE ONE OPERAND DURING THE
588 *                OR OR AND OPERATION.
589 *
590 *        Y       USED AS THE CONTROL VARIABLE IN FOR-NEXT LOOPS.
591 *                ALSO USED AS THE POINTER TO THE H (RESULT) ARRAY
592 *                DURING THE OR ROUTINE.  ALSO USED AS AN INPUT
593 *                VARIABLE IN THE INPUT-MODE CHANGE ROUTINE.
594 *
595 *        Y1      USED AS THE CONTROL VARIABLE IN FOR-NEXT LOOPS.
596 *
597 *        Y5      A PARAMETER INDICATING THE MAXIMUM NUMBER OF TERMS
```

```
598 *              THAT CAN BE STORED IN THE FUNCTION FILE OF EACH
599 *              FUNCTION.
600 *
601 *         Y$   USED AS A TEMPORARY STRING VARIABLE TO STORE THE
602 *              NAME OF A LINE TO BE FAULTED.  ALSO USED AS A
603 *              TEMPORARY VARIABLE TO STORE ONE OPERAND
604 *              DURING THE OR AND THE AND ROUTINES.
605 *
606 *         Z$   USED AS A TEMPORARY STRING VARIABLE TO TRANSFER
607 *              TERMS BETWEEN ARRAYS.  ALSO USED AS AN INPUT
608 *              VARIABLE.  ALSO USED AS A TEMPORARY VARIABLE TO
609 *              STORE THE RESULT TERM DURING THE OR AND THE AND
610 *              ROUTINES.
800*****************************************************************
801*
802*         SUBROUTINE DESCRIPTIONS
803*
804*****************************************************************
805*
809*         PROGRAMMER DEFINED SUBROUTINES
810*
811*    LINE            DESCRIPTION
812*
813*
814*    5000       FUNCTION-FILE TO ARRAY-FILE TRANSFER.
815*
816*    6000       FAULT/CONSTANT-VALUE INSERTION.
817*
818*    6400       MINI-AND.  AND'S FAULTED VARIABLES WITH FF FUNCTION.
819*
820*    6500       MINI-OR.  OR'S FAULTED VARIABLES WITH FF FUNCTION.
821*
822*    7000       OR SUBROUTINE.
823*
824*    7200       TERM SORTING.
825*
826*    7800       PRODUCT TERM DISASSEMBLY INTO LITERAL COMPONENTS.
827*
828*    8000       AND SUBROUTINE.
829*
830*    8250       ZERO PRODUCT TERM DETECTION.
831*
832*    8650       LITERAL SORTING.
833*
834*    9000       ARRAY-FILE TO FUNCTION-FILE TRANSFER.
835*
836*    10000      OLD FUNCTION SET/NEW FUNCTION SET COMPARISON.
837*
838*    11000      NEW FUNCTION SET PRINTING.
839*
840*    13000      OSCILLATION DETECTION.
841*
```

```
842*     17000     FUNCTION MINIMIZATION.
843*
844*     18000     INPUT-MODE CHANGE.
845*
900*****************************************************************************
901*
902*          DIMENSION STATEMENTS
903*
904*****************************************************************************
905*
910       DIM N(200,4), I(400,3), Q(50,2), P(200,2)
920       DIM A$(51), D$(51)
930       DIM U(1)
999*
1000*****************************************************************************
1001*****************************************************************************
1002*                                                                           *
1003*          START OF MAIN PROGRAM                                            *
1004*                                                                           *
1005*****************************************************************************
1006*****************************************************************************
1007*
1008*
1010 OPEN "OLD1" TO :1, INPUT UPDATE
1020 OPEN "OLD0" TO :2, INPUT UPDATE
1030 OPEN "NEW0" TO :3, INPUT UPDATE
1040 OPEN "NEW1" TO :4, INPUT UPDATE
1100 H9=400               \*ASSIGN VALUE TO LOOP LIMIT PARAMETER.
1105 Y5=100               \*ASSIGN FUNCTION MAX LENGTH PARAMETER.
1120 INPUT=6              \*SELECT SEQUENTIAL FILE ACCESS MODE.
1125 IF C0=2 THEN 1450    \*CHECK FOR ENTRY FROM RACE.
1130 S=0                  \*INITIALIZE FUNCTION FILE SWITCH.
1135 PRINT"ENTER INPUT-TIME LIMIT";\*PRINT INPUT-TIME LIMIT
1136                      *ENTRY MESSAGE.
1140 INPUT T0             \*ENTER INPUT-TIME LIMIT.
1145 PRINT"ENTER RIPPLE-TIME LIMIT";\*PRINT RIPPLE-TIME LIMIT
1146                      *ENTRY MESSAGE.
1150 INPUT R0             \*ENTER RIPPLE-TIME LIMIT.
1155 T=1                  \*INITIALIZE INPUT-TIME VARIABLE.
1160 R1=T                 \*SAVE RIPPLE-TIME VALUE AT START OF NEW INPUT-TIME.
1165 R=R1                 \*INITIALIZE RIPPLE-TIME VARIABLE.
1170 FOR I1=1 TO N1       \*BEGIN PRIMARY SIMULATION LOOP.
1175    J=0               \*INITIALIZE FUNCTION TYPE SWITCH.
1180    P0=0              \*INITIALIZE INPUT LIST POINTER INDEX.
1185    K=1               \*INITIALIZE FUNCTION ARRAY SWITCH.
1190    *LOAD FUNCTION ARRAY.\GOSUB 5000
1195    IF I(P(I1,2)+P0,2)="FF" THEN 1215
1196                      *IS THE ELEMENT INPUT FAULTED?
1200    F2=1              \*IF YES, SET FLAG INDICATING FIRST INPUT.
1205    F1=1              \*SET FLAG INDICATING INPUT FAULT.
1210    *INSERT FAULT. \GOSUB 6000
1215    IF N(I1,2)="DELAY" THEN 1320\*CHECK FOR DELAY ELEMENT.
```

```
1217    P0=P0+1
1220    *LOAD FUNCTION ARRAY G.\GOSUB 5000
1225    IF I(P(I1,2)+P0,2)="FF" THEN 1245
1226                    *IS ELEMENT INPUT FAULTED?
1230    F1=1            \*SET INPUT FAULT FLAG.
1235    F2=2            \*SET FLAG INDICATING 2ND OR > INPUT.
1240    *INSERT FAULT. \GOSUB 6000
1245    IF N(I1,2)="DFF" THEN 1317\*CHECK FOR D FLIP-FLOP.
1247    IF J=0 THEN 1270\*IS ZERO-FN BEING COMPUTED?
1250    IF N(I1,2)="NAND" THEN 1260\*ONE-FN IS BEING COMPUTED.
1251                    *CHECK ELEMENT TYPE.
1255    IF N(I1,2)="NOR" THEN 1280
1260    *PERFORM OR OF INPUT  FUNCTIONS.\GOSUB 7000
1265    *PROCEED TO NEXT INPUT IF ANY.\GOTO 1285
1270    IF N(I1,2)="NAND" THEN 1280\*ZERO-FN IS BEING COMPUTED.
1271                    *CHECK ELEMENT TYPE.
1275    IF N(I1,2)="NOR" THEN 1260
1280    *PERFORM AND OF INPUT FUNCTIONS.\GOSUB 8000
1285    IF P(I1,1)-2<P0 THEN 1320\*HAVE ALL ELEMENT INPUTS
1286                    *BEEN PROCESSED?
1290    FOR X=1 TO H9   \*NO.  LOAD F ARRAY WITH PREVIOUS RESULTANT.
1295         INPUT :H;X,Z$\*GET TERM FROM H ARRAY.
1300         PRINT :F;X,Z$\*PLACE TERM IN F ARRAY.
1305         IF Z$="." THEN 1217\*END-OF-FUNCTION?
1310    NEXT X          \* REPEAT FOR NEXT TERM.
1315    *REPEAT FOR NEXT ELEMENT INPUT.\GOTO 1217
1317    *CALL DFF ROUTINE\GOSUB 14000
1318    STOP
1320    IF N(I1,3)="FF" THEN 1335\*ALL INPUTS HAVE BEEN PROCESSED.
1321                    *CHECK FOR FAULTED ELEMENT OUTPUT.
1325    F1=2            \*SET FLAG FOR OUTPUT FAULT.
1330    *INSERT FAULT ON ELEMENT OUTPUT.\GOSUB 6000
1335    *TRANSFER RESULTANT FN TO FN FILE.\GOSUB 9000
1340    IF J<>0 THEN 1355\*HAVE BOTH FNS BEEN COMPUTED?
1345    J=1             \*NO.  SET SWITCH FOR ONE-FUNCTION.
1350    *REPEAT FOR ONE-FUNCTION.\GOTO 1180
1355 NEXT I1            \*REPEAT FOR NEXT ELEMENT.
1360 N0=0               \*ALL ELEMENTS HAVE BEEN PROCESSED.
1361                    *CLEAR FUNCTION SET EQUALITY FLAG.
1365 *CHECK FUNCTION SET EQUALITY.\GOSUB 10000
1370 IF N0=1 THEN 1380 \*ARE THE FUNCTION SETS EQUAL?
1375 *NO.  PROCEED WITH ANALYSIS.\GOTO 1410
1380 R=R-1              \*SUPPRESS RIPPLE-TIME INCREMENT.
1390 IF T>=T0 THEN 1490\*HAS INPUT-TIME LIMIT BEEN REACHED?
1395 T=T+1              \*NO.  INC INPUT-TIME VARIABLE.
1400 R1=R+1             \*SAVE NEW RIPPLE-TIME VALUE.
1405 *CONTINUE SIMULATION.\GOTO 1475
1410 IF N(1,1)="" THEN 1450\*CROSS-COUPLED ELEMENTS?
1415 IF R9=1 THEN 1450 \*RACE ANALYSIS SELECTED?
1420 IF R=1 THEN 1450  \*FIRST RIPPLE STEP?
1425 CLOSE :1           \*CLOSE ALL FILES.
1430 CLOSE :2
```

```
1435 CLOSE :3
1440 CLOSE :4
1445 CHAIN LINK "RACEDOC"  \*PERFORM RACE ANALYSIS PROCEDURE.
1450 IF 41<>2 THEN 1465\*CHECK OUTPUT MODE.
1455 *PRINT LATEST EQUATION SET.\GOSUB 11000
1465 *PERFORM OSCILLATION ANALYSIS PROCEDURE.\GOSUB 13000
1470 IF 02<>1 THEN 1585\*POSSIBLE OSCILLATION?
1475 R=R+1                 \*NO OSCILLATION INDICATED.
1476                       *INC RIPPLE-TIME VARIABLE.
1480 S=1-S                 \*SWITCH FUNCTION FILES.
1485 *CONTINUE FOR NEXT RIPPLE-TIME INCREMENT.\GOTO 1170
1489 *INPUT-TIME LIMIT HAS BEEN REACHED.  INCREASE LIMIT?
1490 PRINT"DO YOU WISH TO ENTER A NEW TIME LIMIT--YES/NO.  <CR>=NO.";
1495 INPUT Z$              \*ENTER RESPONSE.
1500 IF Z$<>"YES" THEN 1585\*IF NO, EXIT PROGRAM.
1505 PRINT"ENTER NEW INPUT-TIME LIMIT";
1510 INPUT T0              \*ENTER NEW INPUT-TIME LIMIT.
1515 PRINT"NEW INPUT MODE--YES/NO.  <CR>=NO.";
1520 INPUT Z$              \*ENTER RESPONSE.
1525 IF Z$<>"YES" THEN 1395
1530 *ENTER NEW INPUT MODE.\GOSUB 18000
1535 *CONTINUE SIMULATION. \GOTO 1395
1545 GOTO 1585
1560 PRINT"ENTER NEW INPUT-TIME LIMIT";
1565 INPUT T0              \*ENTER NEW INPUT-TIME LIMIT.
1570 PRINT"ENTER NEW RIPPLE-TIME LIMIT";
1575 INPUT R0              \*ENTER NEW RIPPLE-TIME LIMIT.
1580 GOTO 1170
1585 CLOSE :1              \*CLOSE ALL FILES.
1590 CLOSE :2
1595 CLOSE :3
1600 CLOSE :4
1601 PRINT "GENERATE TEST SEQUENCES--YES/NO.  <CR>=NO.";
1602 INPUT Z$              \*INPUT RESPONSE TO QUERY.
1603 IF Z$<>"YES" THEN 1605\*DO NOT GENERATE SEQ IF NO.
1604 CHAIN LINK "TESTGNDOC"
1605 CHAIN LINK "SIMDOC"   \*RETURN TO INPUT PROGRAM.
2000*****************************************************************
2001*****************************************************************
2002*          SUBROUTINES                                          *
2003*****************************************************************
5000*****************************************************************
5001*         SUBROUTINE TO TRANSFER FUNCTION FROM
5002*         FUNCTION FILE TO FUNCTION ARRAY.
5003*****************************************************************
5004 INPUT = 0             \*RETURN TO NORMAL INPUT MODE.
5005 F=2-J-S+2*J*S\G=3-S+J-2*J*S\H=F+2\*COMPUTE CHANNEL NUMBERS FOR
5006                       *F, G, AND H ARRAY FILES.
5007 CLOSE :1\CLOSE :2\CLOSE :3\CLOSE :4\*CLOSE ALL OPEN FILES.
5008 IF N(I1,2)="DELAY" THEN 5400\*DELAY ELEMENT?
5009 OPEN "FARRAY" TO :F, INPUT UPDATE\*OPEN FILE FOR F ARRAY.
5010 OPEN "GARRAY" TO :G, INPUT UPDATE\*OPEN FILE FOR G ARRAY.
```

```
5011 OPEN "HARRAY" TO :H, INPUT UPDATE\*OPEN FILE FOR H ARRAY.
5012 IF S=1 THEN 5018  \*DETERMINE WHICH FILE SHOULD BE ACCESSED.
5013 IF J=1 THEN 5016  \*LAST FN SET IN OLD FILES IF S=0.
5014 OPEN "OLD1" TO :1, INPUT UPDATE\*NEED A ONE-FN IF J=0.
5015 *GOTO NEXT STEP   \GOTO 5022
5016 OPEN "OLD0" TO :2, INPUT UPDATE\*NEED A ZERO-FN IF J=1.
5017 *GOTO NEXT STEP.  \GOTO 5022.
5018 IF J=1 THEN 5021  \*LAST FN SET IN NEW FILES IF S=1.
5019 OPEN "NEW1" TO :4, INPUT UPDATE\*NEED A ONE-FN IF J=0.
5020 *GOTO NEXT STEP   \GOTO 5022
5021 OPEN "NEW0" TO :3, INPUT UPDATE\*NEED A ZERO-FN IF J=1.
5022 IF I(P(I1,2)+P0,3)<>"YES" THEN 5135\*IS THE ELEMENT INPUT
5023                    *A PRIMARY INPUT OF THE CIRCUIT?
5029 *YES.  THEREFORE, THE INPUT FUNCTION MUST BE CONSTRUCTED.
5030 U(1)=T+240          \*COMPUTE EBCDIC EQUIVALENT OF INPUT-TIME.
5040 CHANGE U TO T$      \*CONVERT INPUT-TIME TO STRING VARIABLE.
5050 IF J<>0 THEN 5080\*FORM ZERO-INPUT OR ONE-INPUT?
5060 Z$=""               \*ZERO-FN NEEDS A ONE-INPUT.
5070 *CONTINUE TO NEXT STEP\GOTO 5090
5080 Z$="-"              \*ONE-FN NEEDS A ZERO-INPUT.
5090 IF K<>1 THEN 5120 \*WHICH ARRAY IS TO BE LOADED?
5100 X$=I(P(I1,2)+P0,1)+Z$+T$\*CONSTRUCT INPUT EXPRESSION.
5102 PRINT :F;1,X$     \*PLACE EXPRESSION IN F ARRAY.
5105 PRINT :F;2,"."    \*LOAD END-OF-FUNCTION MARK.
5110 *CONTINUE TO NEXT STEP\GOTO 5300
5120 Y$=I(P(I1,2)+P0,1)+Z$+T$\*CONSTRUCT INPUT EXPRESSION.
5122 PRINT :G;1;Y$     \*LOAD EXPRESSION IN G ARRAY.
5125 PRINT :G;2,"."\*LOAD END-OF-FUNCTION MARK.
5130 *CONTINUE TO NEXT STEP\GOTO 5300
5134 *ELEMENT INPUT IS NOT A PRIMARY INPUT.  GET FN FROM FILE.
5135 J$=I(P(I1,2)+P0,3)      \*GET INPUT ID NUMBER.
5140 V=VAL(J$)               \*CONVERT ID TO A NUMERIC VARIABLE.
5150 INPUT :J=2*S*J+3*S+1;Y5*(V-1),X$\*RETRIEVE FN NAME FROM FILE.
5160 FOR X=1 TO 1000         \*TRANSFERS TERMS OF FN FROM FILE TO ARRAY
5170    INPUT :J=2*S*J+3*S+1,Y$\*GET TERM FROM FILE.
5180    PRINT :G;X,Y$        \*PLACE TERM IN ARRAY.
5190    IF Y$="." THEN 5260 \*LOOK FOR END OF FUNCTION.
5200 NEXT X                  \*REPEAT FOR NEXT TERM.
5260 IF K<>1 THEN 5300       \*LOAD F ARRAY?
5270 FOR Y=1 TO X            \*YES.  TRANSFER FROM G TO F.
5280    INPUT :G;Y,Y$        \*GET TERM FROM G.
5285    PRINT :F;Y,Y$        \*PLACE TERM IN F.
5290 NEXT Y                  \*REPEAT FOR NEXT TERM.
5300 K=0                     \*CLEAR THE ARRAY SWITCH.
5305 INPUT=$                 \*SEQUENTIAL FILE INPUT MODE.
5310 RETURN                  \*RETURN TO MAIN PROGRAM.
5400 H1=I+J+S*S-2*S*J  \*DEFINE CHANNEL FUNCTION FOR HARRAY.
5402 OPEN "HARRAY" TO :H1,INPUT UPDATE
5403 IF S=1 THEN 5440  \*CHECK FUNCTION-FILE SWITCH.
5405 IF J=1 THEN 5425  \*CHECK FUNCTION FLAG.
5410 OPEN "OLD0" TO :2,INPUT UPDATE
5420 GOTO 5500
```

```
5425 OPEN "OLD1" TO :1,INPUT UPDATE
5435 GOTO 5500
5440 IF J=1 THEN 5460  \*CHECK FUNCTION FLAG.
5445 OPEN "NEW0" TO :3,INPUT UPDATE
5455 GOTO 5500
5460 OPEN "NEW1" TO :4,INPUT UPDATE
5500 IF I(P(I1,2),3)<>"YES" THEN 5560\*CHECK FOR PRIMARY INPUT.
5505 U(1)=T+240          \*COMPUTE EBCDIC OF TIME.
5510 CHANGE U TO T$      \*CONVERT TIME TO STRING.
5515 IF J<>0 THEN 5530 \*CHECK FOR 0-FN FORMATION.
5520 Z$="-"              \*FORM - INPUT VARIABLE.
5525 GOTO 5535
5530 Z$=""               \*FORM INPUT VARIABLE.
5535 Y$=I(P(I1,2),1)+Z$+T$\*COMPLETE VARIABLE FORMATION.
5540 PRINT :H1;1,Y$      \*LOAD FUNCTION.
5545 PRINT :H1;2,"."
5550 INPUT=$             \*CHANGE INPUT MODE.
5555 RETURN
5560 J$=I(P(I1,2),3)     \*GET INPUT NAME INDEX.
5565 V=VAL(J$)           \*CONVERT INDEX TO NUMERIC.
5570 INPUT :2+S-J+2*S*J;Y$*(V-1),X$\*GET INPUT FUNCTION.
5575 FOR X=1 TO 1000     \*LOAD TERMS OF INPUT FUNCTION.
5580    INPUT :2+S-J+2*S*J,Y$
5585    PRINT :H1;X,Y$
5590    IF Y$="." THEN 5600
5595 NEXT X
5600 INPUT=$             \*CHANGE INPUT MODE.
5604 *CLOSE ALL FILES.
5605 CLOSE :1\ CLOSE :2\ CLOSE :3\ CLOSE :4
5609 *REOPEN FILES FOR NEXT PROCESSING STEP.
5610 OPEN "FARRAY" TO :F,INPUT UPDATE
5615 OPEN "GARRAY" TO :G,INPUT UPDATE
5620 OPEN "HARRAY" TO :H,INPUT UPDATE
5625 IF S=1 THEN 5640
5630 IF J=1 THEN 5635
5631 OPEN "OLD1" TO :1,INPUT UPDATE
5632 GOTO 5700
5635 OPEN "OLD0" TO :2,INPUT UPDATE
5637 GOTO 5700
5640 IF J=1 THEN 5650
5645 OPEN "NEW1" TO :4,INPUT UPDATE
5647 GOTO 5700
5650 OPEN "NEW0" TO :3,INPUT UPDATE
5700 RETURN
6000*************************************************************************
6001*              SUBROUTINE FOR INSERTING
6002*              FAULTS AND CONSTANT INPUTS.
6003*************************************************************************
6010 IF F1=2 THEN 6800 \*CHECK FOR OUTPUT FAULT (F1=2).
6020 X$=I(P(I1,2)+P0,2)\*GET INPUT FAULT OR VALUE.
6030 Y$=I(P(I1,2)+P0,1)\*GET INPUT LINE NAME.
6035 IF I(I1,2)="DELAY" THEN 6600\*IS ELEMENT A DELAY ELEMENT.
```

```
6040 IF F2=2 THEN 6070 \*IS THE INPUT THE SECOND OR HIGHER?
6050 G1=F              \*THE FIRST INPUT.  USE ARRAY F.
6060 *CONTINUE TO NEXT STEP.\GOTO 6100
6070 G1=G              \*YES.  USE ARRAY G.
6100 IF X$="S0" THEN 6150\*IDENTIFY FAULT OR CONSTANT VALUE.
6110 IF X$="S1" THEN 6200\*TRANSFER CONTROL TO PROPER ROUTINE.
6120 IF X$="0" THEN 6250
6130 IF X$="1" THEN 6300
6140 PRINT"INVALID FAULT"
6145 STOP
6150 IF J=0 THEN 6180  \*CHECK FUNCTION FLAG.
6160 Y$=Y$+"*"         \*INSERT STUCK-AT-ZERO SYMBOL.
6165 *OR FAULT CONDITION WITH FF FUNCTION.\GOSUB 6500
6170 RETURN            \*RETURN TO MAIN PROGRAM.
6180 Y$=Y$+"-*"        \*INSERT NOT STUCK-AT-ZERO SYMBOL.
6185 *AND FAULT CONDITION WITH FF FUNCTION.\GOSUB 6400
6190 RETURN            \*RETURN TO MAIN PROGRAM.
6200 IF J=0 THEN 6230\*CHECK FUNCTION FLAG.
6210 Y$=Y$+"-!"        \*INSERT NOT STUCK-AT-ONE SYMBOL.
6215 *AND FAULT CONDITION WITH FF FUNCTION.\GOSUB 6400
6220 RETURN            \*RETURN TO MAIN PROGRAM.
6230 Y$=Y$+"!"         \*INSERT STUCK-AT-ONE SYMBOL.
6235 *OR THE FAULT CONDITION WITH FF FUNCTION.\GOSUB 6500
6240 RETURN            \*RETURN TO MAIN PROGRAM.
6250 IF J=1 THEN 6280  \*CHECK FUNCTION FLAG.
6260 PRINT :G1;1,"1"   \*LOAD INPUT VALUE OF ONE.
6265 PRINT :G1;2,"."   \*LOAD END-OF-FUNCTION MARK.
6270 RETURN            \*RETURN TO MAIN PROGRAM.
6280 PRINT :G1;1,"0"   \*LOAD INPUT VALUE OF ZERO.
6285 PRINT :G1;2,"."   \*LOAD END-OF-FUNCTION MARK.
6290 RETURN            \*RETURN TO MAIN PROGRAM.
6300 IF J=1 THEN 6260  \*CHECK FUNCTION FLAG.
6310 *J=0,             \GOTO 6280
6400*************************************************************************
6401*         SUBROUTINE TO AND FAULTED VARIABLES WITH FF FUNCTION.
6402*         FF FUNCTION MAY BE IN F, G, OR H ARRAY.
6403*         G1 MUST BE ASSIGNED THE VALUE SPECIFYING
6404*         WHICH ARRAY CONTAINS THE FF FUNCTION.
6405*************************************************************************
6410 FOR Y1=1 TO H9    \*AND FAULTED VARIABLE WITH EACH TERM OF FN.
6420     INPUT :G1;Y1,Z$\*GET TERM OF FUNCTION.
6425     IF Z$="0" THEN 6470\*CHECK FOR ZERO FUNCTIONAL VALUE.
6430     IF Z$="." THEN 6470\*CHECK FOR END OF FUNCTION.
6435     IF Z$<>"1" THEN 6440\*CHECK FOR ONE FUNCTIONAL VALUE.
6436     Z$=Y$          \*IF Z$=1 THEN LET Z$=FAULTED VARIABLE.
6437     *CONTINUE TO NEXT STEP\GOTO 6450
6440     Z$=Z$+Y$       \*IF Z$<>0 THEN CONCATENATE THE TERMS.
6450     PRINT :G1;Y1,Z$\*STORE THE MODIFIED TERM.
6460 NEXT Y1            \*REPEAT FOR NEXT TERM.
6470 RETURN             \*RETURN TO FAULT INSERTION SUBROUTINE.
6500*************************************************************************
6501*         SUBROUTINE TO OR FAULTED VARIABLES WITH FF FUNCTION.
```

```
6502*         FF FUNCTION MAY BE IN F, G, OR H ARRAY.
6503*         G1 MUST BE LOADED WITH THE VALUE OF F, G, OR H.
6504*****************************************************************
6510 FOR Y1=1 TO H9      \*SCAN TERMS OF FF FUNCTION.
6520    INPUT :G1;Y1,Z$\*GET TERM FROM ARRAY.
6525    IF Z$="0" THEN 6550\*LOOK FOR ZERO FUNCTIONAL VALUE.
6527    IF Z$="1" THEN 6570\*LOOK FOR ONE FUNCTIONAL VALUE.
6530    IF Z$="." THEN 6550\*LOOK FOR END OF FUNCTION.
6540 NEXT  Y1            \*REPEAT FOR NEXT TERM.
6550 PRINT :G1;Y1,Y$     \*INSERT FAULTED VARIABLE AS NEW TERM
6551                     *OF NON-ZERO FUNCTION.
6560 PRINT :G1;Y1+1,"."\*LOAD END-OF-FUNCTION MARK.
6570 RETURN              \*RETURN TO FAULT INSERTION SUBROUTINE.
6600 G1=H                    \*DEFINE CHANNEL FOR HARRAY.
6601 IF X$="S0" THEN 6630    \*STUCK-AT-0 FAULT?
6605 IF X$="S1" THEN 6665    \*STUCK-AT-1 FAULT?
6610 IF X$="0" THEN 6700     \*CONSTANT 0 VALUE?
6615 IF X$="1" THEN 6735     \*CONSTANT 1 VALUE?
6620 PRINT "INVALID FAULT."
6625 STOP
6630 IF J=0 THEN 6650        \*CHECK FUNCTION FLAG.
6635 Y$=Y$+"-*"              \*INSERT -FAULT(S0) SYMBOL.
6640 *INSERT FAULT IN FUNCTION.\GOSUB 6400
6645 RETURN
6650 Y$=Y$+"*"               \*INSERT S0 FAULT SYMBOL.
6655 *INSERT FAULT IN FUNCTION.\GOSUB 6500
6660 RETURN
6665 IF J=0 THEN 6685        \*CHECK FUNCTION FLAG.
6670 Y$=Y$+"!"               \*INSERT S1 FAULT SYMBOL.
6675 *INSERT FAULT IN FUNCTION\GOSUB 6500
6680 RETURN
6685 Y$=Y$+"-!"              \*INSERT S1- FAULT SYMBOL.
6690 *INSERT FAULT IN FUNCTION\GOSUB 6400
6695 RETURN
6700 IF J=1 THEN 6720        \*CHECK FUNCTION FLAG.
6705 PRINT :F;1,"1"          \*LOAD CONSTANT 1.
6710 PRINT :F;2,"."
6715 RETURN
6720 PRINT :F;1,"0"          \*LOAD CONSTANT 0.
6725 PRINT :F;2,"."
6730 RETURN
6735 IF J=1 THEN 6705        \*CHECK FUNCTION FLAG.
6740 GOTO 6720
6800 X$=N(I1,5)          \*GET FAULT CONDITION.
6802 Y$=N(I1,1)          \*GET OUTPUT LINE NAME.
6804 G1=H                \*FF FUNCTION IS IN H ARRAY.
6807 IF X$="S0" THEN 6850\*IDENTIFY FAULT CONDITION.
6810 IF X$="S1" THEN 6860\*AS FIRST STEP IN TRANSFER TO
6820 IF X$="0" THEN 6870 \*PROPER PROCESSING ROUTINE.
6830 IF X$="1" THEN 6880
6840 GOTO 6140
6849 *CHECK FUNCTION FLAG (J) FOR FINAL ROUTINE SELECTION.
```

```
6850 IF J=0 THEN 6160
6855 GOTO 6180
6860 IF J=0 THEN 6210
6865 GOTO 6230
6870 IF J=1 THEN 6280
6875 GOTO 6260
6880 IF J=1 THEN 6260
6885 GOTO 6280
7000*****************************************************************
7001*        SUBROUTINE FOR ORING TWO FUNCTIONS.
7002*   ONE FUNCTION MUST BE IN ARRAY F AND THE SECOND IN ARRAY G.
7003*   THE RESULTANT FUNCTION IS PLACED IN ARRAY H.
7004*   CALLED BY THE MAIN PROGRAM ONLY.
7005*   CALLS THE SORT SUBROUTINE AND THE MINIMIZATION SUBROUTINE.
7006*****************************************************************
7010 INPUT :F;1,X$      \*GET FIRST TERM OF FUNCTION FROM F.
7011 IF X$<>"0" THEN 7035\*CHECK FOR ZERO FUNCTIONAL VALUE.
7015 INPUT :G;1,Y$      \*GET FIRST TERM OF FUNCTION FROM G.
7016 IF Y$<>"0" THEN 7075\*CHECK FOR ZERO FUNCTIONAL VALUE.
7020 PRINT :H;1,"0"     \*BOTH INPUTS ARE ZERO.  RESULT IS ZERO.
7025 PRINT :H;2,"."     \*LOAD END-OF-FUNCTION MARK.
7030 RETURN             \*RETURN TO MAIN PROGRAM.
7035 INPUT :G;1,Y$      \*GET FIRST TERM OF FUNCTION FROM G.
7036 IF Y$<>"0" THEN 7110\*FIRST INPUT NOT ZERO, IS THE SECOND?
7040 FOR X=1 TO H9      \*SECOND INPUT IS ZERO.
7045    INPUT :F;X,X$\*RESULTANT IS EQUAL TO FIRST FUNCTION.
7046    PRINT :H;X,X$
7050    IF X$="." THEN 7060\*LOOK FOR END OF FUNCTION.
7055 NEXT X             \*REPEAT FOR NEXT TERM.
7060 *SORT THE TERMS.  \GOSUB 7200
7065 *REDUCE THE RESULTANT FN.\GOSUB 17000
7070 RETURN             \*RETURN TO MAIN PROGRAM.
7075 FOR X=1 TO H9      \*FIRST FN=0.  SECOND FN<>0.
7080    INPUT :G;X,Y$   \*RESULTANT FN = SECOND FN.
7081    PRINT :H;X,Y$
7085    IF Y$="." THEN 7095\*LOOK FOR END OF FUNCTION.
7090 NEXT X             \*REPEAT FOR NEXT TERM.
7095 *SORT THE TERMS.  \GOSUB 7200
7100 *REDUCE THE RESULTANT FN.\GOSUB 17000
7105 RETURN             \*RETURN TO MAIN PROGRAM.
7110 INPUT :F;1,X$      \*GET FIRST TERM OF FIRST FN.
7111 IF X$<>"1" THEN 7130\*CHECK FOR ONE FUNCTIONAL VALUE.
7115 PRINT :H;1,"1"     \*FIRST FUNCTION IS ONE. RESULT IS ONE.
7120 PRINT :H;2,"."     \*LOAD END-OF-FILE MARK.
7125 RETURN             \*RETURN TO MAIN PROGRAM.
7130 INPUT :G;1,Y$      \*GET FIRST TERM OF SECOND FUNCTION.
7131 IF Y$<>"1" THEN 7150\*CHECK FOR ONE FUNCTIONAL VALUE.
7135 PRINT :H;1,"1"     \*SECOND FN = 1.  RESULTANT FN =1.
7140 PRINT :H;2,"."     \*LOAD END-OF-FUNCTION MARK.
7145 RETURN             \*RETURN TO MAIN PROGRAM.
7149 *NEITHER INPUT FN IS ZERO OR ONE.  COMPUTE THE RESULT.
7150 Y=1                \*INITIALIZE POINTER TO H ARRAY.
```

```
7155 FOR X=1 TO H9      \*COPY FIRST FN INTO H ARRAY.
7160    INPUT :F;X,X$   \*GET TERM OF FN.
7161    PRINT :H;Y,X$   \*PLACE TERM IN H.
7165    INPUT :F;X+1,X$\*GET NEXT TERM OF FN.
7166    IF X$="." THEN 7180\*END OF FN?
7170    Y=Y+1           \*NO.  INC POINTER TO H.
7175 NEXT X             \*INC POINTER TO F.
7180 Y=Y+1              \*YES.  INC POINTER TO H.
7185 FOR X=1 TO H9      \*COPY G INTO H FOLLOWING F.
7187    INPUT :G;X,Y$   \*GET TERM OF G.
7188    PRINT :H;Y,Y$   \*PLACE TERM IN H.
7189    IF Y$="." THEN 7195\*LOOK FOR END OF FN G.
7191    Y=Y+1           \*END NOT REACHED.  INC POINTER TO H.
7193 NEXT X             \*INC POINTER TO G.
7195 *SORT TERMS.       \GOSUB 7200
7197 *REDUCE RESULTANT FUNCTION.\GOSUB 17000
7199 RETURN             \*RETURN TO MAIN PROGRAM.
7200*************************************************************************
7201*        SUBROUTINE TO SORT THE TERMS OF A FUNCTION.
7202*        FUNCTION MUST BE STORED IN ARRAY H BEFORE
7203*        SORTING RULES: LENGTH OF TERM, ALPHABETICAL, NUMERICAL.
7204*        CALLED BY OR AND AND SUBROUTINES.
7205*        CALLS NO SUBROUTINES.
7206*************************************************************************
7220 FOR X=1 TO 100      \*COUNT NUMBER OF TERMS IN FN.
7230    INPUT :H;X,Z$    \*GET A TERM.
7231    IF Z$="." THEN 7250 \*END OF FUNCTION?
7240 NEXT X              \*NO.
7249 *X = THE NUMBER OF TERMS IN H PLUS 1.
7250 IF X=2 THEN 7500    \*CHECK FOR SINGLE TERM FN.
7252 X0=X                \*SAVE X FOR USE AS SORT LOOP LIMITER.
7255 X1=X                \*SAVE X FOR STORE LOOP LIMITER.
7260 FOR X=1 TO X0-2     \*USE A BUBBLE SORT TO ORDER THE TERMS.
7270    INPUT :H;X,X$    \*GET FIRST TERM OF H.
7272    INPUT :H;X+1,Y$ \*GET ADJACENT TERM OF H.
7274    IF X$<Y$ THEN 7310\*COMPARE THE TERMS.
7280    *REVERSE THE ORDER OF THE TWO TERMS.
7290    PRINT :H;X,Y$
7300    PRINT :H;X+1,X$
7310 NEXT X              \*REPEAT FOR NEXT PAIR OF TERMS.
7319 * ONE TERM HAS BEEN PLACED IN ITS PROPER PLACE.
7320 X0=X0-1             \*DECREMENT SORT LOOP LIMITER.
7330 IF X0>2 THEN 7260   \*HAVE ALL TERMS BEEN ORDERED?
7340 X0=X1               \*YES.  RESTORE X0.
7344 *DUPLICATE TERMS WILL NOW BE REMOVED.
7345 FOR X=1 TO X1       \*COMPARE ADJACENT TERMS FOR EQUALITY.
7347    INPUT :H;X,X$    \*GET A TERM.
7348    IF X$="." THEN 7420\*END OF FUNCTION?
7350    INPUT :H;X+1,Y$\*NO.  GET ADJACENT TERM.
7351    IF X$=Y$ THEN 7360\*ARE TERMS IDENTICAL?
7355    *NO.  GET NEXT PAIR.\GOTO 7410
7360    X0=X0-1          \*YES.  DECREMENT TERM COUNT.
```

```
7365     FOR Y=X+1 TO X1-1\*REMOVE DUPLICATE TERM AND COMPACT H.
7370          INPUT :H;Y+1,X$\*GET TERM FROM H.
7371          PRINT :H;Y,X$\*MOVE TERM TO NEW POSITION IN H.
7380          IF X$="," THEN 7410\*END OF FUNCTION?
7390     NEXT Y                \*REPEAT FOR NEXT TERM.
7410 NEXT X                    \*REPEAT FOR NEXT PAIR OF TERMS.
7420 FOR X=1 TO X0-2           \*SORT TERMS BY LENGTH USING BUBBLE SORT.
7430     INPUT :H;X,X$
7431     INPUT :H;X+1,Y$
7432     L1=LEN(X$)
7435     L2=LEN(Y$)
7434     IF L1<=L2 THEN 7470\*IF TRUE, DO NOT REORDER.
7440     PRINT :H;X,Y$         \*REORDER.
7450     PRINT :H;X+1,X$
7470 NEXT X                    \*REPEAT FOR NEXT PAIR.
7480 X0=X0-1                   \*DECREMENT LOOP LIMIT FOR NEXT PASS.
7490 IF X0>2 THEN 7420         \*SORTING COMPLETE?
7500 RETURN                    \*YES.  RETURN TO CALLING PROGRAM.
7800*************************************************************
7801*        SUBROUTINE TO DISASSEMBLE A PRODUCT TERM INTO ITS
7802*   LITERAL COMPONENTS.  A LITERAL IS A VARIABLE NAME OR A
7803*   COMPLEMENTED VARIABLE NAME TOGETHER WITH ITS TIME TAG
7804*   OR FAULT SYMBOL.
7805*   TERM IS PASSED TO SUBPOUTINE IN X$.  LITERALS RETURNED
7806*   IN A$ ARRAY.  CALLED BY LITERAL SORT SUBROUTINE.
7807*************************************************************
7810 F5=0                      \*CLEAR SPECIAL CHARACTER FLAG.
7815 Z$=""                     \*CLEAR DUMMY STRING VARIABLE.
7820 A=1                       \*INITIALIZE POINTER TO A$ ARRAY.
7830 L=LEN(X$)                 \*COMPUTE LENGTH OF TERM.
7835 FOR X3=1 TO L             \*BEGIN DISASSEMBLY OF TERM.
7840     IF X$(:X3,1)="-" THEN 7900\*LOOK FOR SPECIAL CHARS (-).
7842     IF X$(:X3,1)>"Z" THEN 7900\*LOOK FOR TIME TAG (NUMERAL).
7844     IF X$(:X3,1)<"A" THEN 7900\*LOOK FOR FAULT SYMBOLS (!/*).
7850     IF F5<>1 THEN 7910\*END OF LITERAL REACHED?
7860     A$(A)=Z$              \*YES.  PLACE LITERAL IN A$ ARRAY.
7861                            *END OF LITERAL IS INDICATED WHEN A SPECIAL
7862                            *CHARACTER IS FOLLOWED BY A LETTER.
7863                            *THE LETTER IS THE FIRST CHARACTER OF THE
7864                            *NEXT LITERAL.
7870     Z$=X$(:X3,1)          \*BEGIN CONSTRUCTING NEXT LITERAL.
7880     F5=0                  \*CLEAR SPECIAL CHARACTER FLAG.
7890     A=A+1                 \*INC POINTER TO A$ ARRAY.
7895     GOTO 7920
7896                            *CONTINUE PROCESSING TERM.
7900     F5=1                  \*SET SPECIAL CHARACTER FLAG.
7910     Z$=Z$+X$(:X3,1)       \*CONSTRUCT LITERAL.
7920 NEXT X3                   \*REPEAT FOR NEXT CHARACTER OF TERM.
7930 A$(A)=Z$                  \*LOAD LAST LITERAL OF TERM IN A$ ARRAY.
7940 A$(A+1)="."               \*LOAD END-OF-LITERAL MARK.
7950 RETURN                    \*RETURN TO LITERAL SORT SUBROUTINE.
8000*************************************************************
```

```
8001*          SUBROUTINE TO PERFORM THE LOGICAL AND OF
8002*          TWO FUNCTIONS.
8003*          THE INPUT FUNCTIONS ARE PASSED IN ARRAYS F AND G.
8004*     THE RESULTANT FUNCTION IS RETURNED IN ARRAY H.
8005*          CALLED BY MAIN PROGRAM (SIM2) AND RACE PROGRAM.
8006*          CALLS ZERO PRODUCT DETECT SUBR (8250), LITERAL SORT
8007*     SUBR(8650), TERM SORT SUBR(7200), AND MINIMIZATION SUBR(17000)
8008*****************************************************************
8010 INPUT :F;1,X$              \*CHECK FOR ZERO FN VALUE.
8011 IF X$="0" THEN 8030
8020 INPUT :G;1,Y$              \*CHECK FOR ZERO FN VALUE.
8021 IF Y$<>"0" THEN 8050
8030 PRINT :H;1,"0"             \*RESULT IS ZERO.
8040 PRINT :H;2,"."
8045 RETURN                     \*RETURN TO CALLING PROGRAM.
8050 IF X$="1" THEN 8058        \*CHECK FOR ONE VALUED FUNCTION.
8051 IF Y$<>"1" THEN 8068       \*CHECK FOR ONE VALUED FUNCTION.
8052 FOR X=1 TO H9              \*2ND FN ONLY IS ONE VALUED.  RESULT = F.
8053    INPUT :F;X,X$            \*GET TERM OF F.
8054    PRINT :H;X,X$            \*PLACE IN H ARRAY.
8055    IF X$="." THEN 8160      \*END OF FUNCTION?
8056 NEXT X                     \*REPEAT FOR NEXT TERM.
8057 *PROCEED TO FUNCTION REDUCTION STEP.\GOTO 8160
8058 IF Y$<>"1" THEN 8062\*1-ST FN IS 1-VALUED. IS THE 2ND FN 1-VALUED?
8059 PRINT :H;1,"1"             \*BOTH FNS ARE 1-VALUED.  RESULT IS 1-VALUED.
8060 PRINT :H;2,"."
8061 RETURN                     \*RETURN TO CALLING PROGRAM.
8062 FOR X=1 TO H9              \*1ST FN IS 1-VALUED.  2ND IS NOT.
8063    INPUT :G;X,Y$            \*RESULT = G.
8064    PRINT :H;X,Y$
8065    IF Y$="." THEN 8160\*END OF FUNCTION?
8066 NEXT X
8067 *PROCEED TO MINIMIZATION STEP.\GOTO 8160
8068 W=1              \*NEITHER INPUT IS CONSTANT-VALUED. INIT POINTER TO H.
8069 PRINT :H;1,"0"             \*INITIALIZE H TO 0-VALUE.
8070 PRINT :H;2,"."
8075 FOR X=1 TO H9              \*BEGIN PRODUCT-TERM FORMATION LOOP.
8080    INPUT :F;X,X$           \*GET TERM FROM F.
8090    FOR Y=1 TO H9           \*BEGIN LOOP FOR ACCESS TO G.
8091       INPUT :G;Y,Y$\*GET TERM FROM G.
8092       Z$=X$+Y$             \*FORM PRODUCT OF TERMS.
8093       PRINT :H;W,Z$\*PLACE RESULT IN H.
8100       *CHECK FOR ZERO PRODUCT.\GOSUB 8250
8110       INPUT :G;Y+1,Y$\*CHECK G FOR END OF FUNCTION.
8111       IF Y$="." THEN 8130
8120    NEXT Y                  \*NOT END OF G.  GET NEXT TERM.
8130    INPUT :F;X+1,X$\*CHECK F FOR END OF FUNCTION.
8131    IF X$="." THEN 8150
8140 NEXT X                     \*NOT END OF F.  GET NEXT TERM.
8150 IF W<>1 THEN 8155 \*END OF BOTH FNS REACHED.  0-VALUE RESULT?
8152 PRINT :H;1,"0"             \*YES.  LOAD 0-VALUE IN H.
8153 RETURN                     \*RETURN TO CALLING PROGRAM.
```

```
8155 PRINT :H;W,"."     \*NO.  LOAD END-OF-FUNCTION MARK.
8160 FOR X=1 TO H9      \*BEGIN MIN STEP BY ORDERING LITS IN EACH TERM.
8165    INPUT :H;X,X$   \*GET A TERM FROM H.
8167    IF Y$="." THEN 8190\*END OF FUNCTION?
8170    *NO.  ORDER LITERALS.\GOSUB 8650
8175    PRINT :H;X,Z$   \*REPLACE ORDERED TERM.
8180 NEXT X             \*REPEAT FOR NEXT TERM.
8190 *ORDER TERMS OF THE FUNCTION.\GOSUB 7200
8200 *REMOVE REDUNDANT TERMS.\GOSUB 17000
8210 RETURN             \*RETURN TO CALLING PROGRAM.
8250*************************************************************
8251*        SUBROUTINE TO CHECK FOR ZERO PRODUCT TERM.
8252*     TERM TO BE CHECKED WAS FORMED BY THE AND SUBROUTINE
8253*   AND WAS PLACED IN THE H ARRAY.  THE TERM IS RETRIEVED FROM H
8254*   FOR PROCESSING.  IF THE TERM IS ZERO, THE POINTER (W) TO THE
8255*   H ARRAY IS NOT INCREMENTED WHICH RESULTS IN THE TERM BEING
8256*   OVERWRITTEN BY ANOTHER TERM. IF THE TERM IS NON-0, W IS INC.
8257*        CALLED BY THE AND SUBROUTINE.
8258*************************************************************
8260 C=1                \*SET THE SPECIAL CHARACTER FLAG.
8270 D=1                \*SET THE MINUS FLAG.
8280 Z$=""              \*CLEAR A TEMPORARY VARIABLE.
8290 B$=""              \*CLEAR ANOTHER TEMPORARY VARIABLE.
8300 INPUT :H;W,Y$      \*GET TERM TO BE ANALYZED.
8301 L=LEN(Y$)          \*COMPUTE  LENGTH OF TERM.
8310 FOR A=1 TO L       \*BEGIN TERM ANALYSIS.
8320    X$=Y$(:A,1)     \*EXAMINE A CHARACTER FROM Y$.
8330    IF X$>"Z" THEN 8350\*NUMERAL?
8332    IF X$="-" THEN 8350\*MINUS SIGN?
8334    IF X$="*" THEN 8350\*STUCK-AT-0 SYMBOL?
8336    IF X$="!" THEN 8350\*STUCK-AT-1 SYMBOL?
8340    IF C=1 THEN 8400\*END OF LITERAL?
8341    *YES.  LOOK FOR ITS COMPLEMENT.\GOTO 8420
8350    C=0             \*SPECIAL SYMBOL FOUND.  CLEAR FLAG.
8360    IF X$<>"-" THEN 8370\*WAS THE SYMBOL A - ?
8365    D=0             \*YES.  CLEAR FLAG.
8366    *GET NEXT CHARACTER.\GOTO 8600
8370    IF D<>1 THEN 8400\*CHECK FOR COMPLEMENTED LITERAL.
8390    Z$=Z$+"-"       \*NOT COMPLEMENTED.  INSERT MINUS.
8400    Z$=Z$+X$        \*CONSTRUCT TEST LITERAL.
8410    *GET NEXT CHARACTER.\GOTO 8600
8420    C=1                     \*SET SPECIAL CHARACTER FLAG.
8430    B1=A                    \*SAVE POINTER TO Y$.
8440    FOR B=B1 TO L           \*FIND NEXT LITERAL IN TERM.
8450       INPUT :H;W,Y$        \*GET THE TERM FROM THE FILE.
8451       X$=Y$(:B,1)          \*GET A CHARACTER OF THE TERM.
8460       IF X$>"Z" THEN 8480\*A NUMERAL?
8462       IF X$="-" THEN 8480\*A MINUS SIGN?
8464       IF X$="*" THEN 8480\*STUCK-AT-0 SYMBOL?
8466       IF X$="!" THEN 8480\*STUCK-AT-1 SYMBOL?
8470       IF C=1 THEN 8490\*END OF LITERAL?
8471       *YES.   COMPARE TO TEST LITERAL.\GOTO 8510
```

```
8480          C=0                \*SPECIAL CHARACTER. CLEAR FLAG.
8490          B$=B$+X$           \*CONSTRUCT LITERAL.
8500          *GET NEXT CHARACTER.\*GOTO 8540
8510          IF Z$=B$ THEN 8620\*COMPARE WITH TEST LITERAL.
8520          C=1                \*NOT = .  SET SPECIAL CHAR FLAG.
8530          B$=""              \*CLEAR TEMPORARY VARIABLE.
8535          *SEARCH FOR NEXT LITERAL.\GOTO 8450
8540      NEXT B                 \*REPEAT FOR NEXT CHARACTER.
8550      IF Z$=B$ THEN 8620     \*COMPARE LAST LITERAL WITH TEST.
8560      C=1                    \*SET SPECIAL CHARACTER FLAG.
8570      D=1                    \*SET THE MINUS SIGN FLAG.
8580      Z$=""                  \*CLEAR A TEMPORARY VARIABLE.
8590      B$=""                  \*CLEAR ANOTHER TEMP VARIABLE.
8595      *DEVELOP A NEW TEST LITERAL.\GOTO 8520
8600  NEXT A                     \*REPEAT FOR NEXT CHARACTER.
8610  N=N+1                      \*NON-0 TERM.  INC POINTER TO N.
8620  RETURN                     \*RETURN TO AND SUBROUTINE.
8650*****************************************************************
8651*         SUBROUTINE TO SORT THE LITERALS IN A PRODUCT TERM.
8652*    LITERALS ARE PASSED TO SUBROUTINE IN Z$ AND THE SORTED
8653*    TERM IS RETURNED IN Z$.
8654*         CALLED BY THE AND SUBROUTINE.
8655*         CALLS THE TERM DISASSEMBLY SUBROUTINE.
8656*****************************************************************
8660  *DISASSEMBLE THE TERM INTO LITERALS.\GOSUB 7800
8665  C=0                        \*INITIALIZE COUNTER.
8668  B1=A                       \*GET NUMBER OF LITERALS IN TERM FROM SUBR 7800
8670  FOR A=1 TO B1              \*SORT THE LITERALS IN A$ USING A BUBBLE SORT.
8680      IF A$(A+1)="." THEN 8740\*END OF LITERALS.
8690      IF A$(A)<A$(A+1) THEN 8730\*COMPARE TWO LITERALS.
8700      Z$=A$(A)               \*RE-ORDERING NEEDED.  SAVE ONE LITERAL.
8710      A$(A)=A$(A+1)          \*SWAP PLACES OF THE TWO.
8720      A$(A+1)=Z$             \*COMPLETE THE SWAP.
8730  NEXT A                     \*REPEAT FOR NEXT PAIR.
8740  IF B1=1 THEN 8750          \*CHECK FOR END OF SORT.
8745  B1=B1-1                    \*DECREASE LOOP LIMIT.  CONTINUE SORT.
8746  *REPEAT FOR NEXT PASS THROUGH THE LIST.\GOTO 8670
8750  FOR A=1 TO 20              \*SORT COMPLETE.  LOOK FOR DUPLICATE LITERALS.
8760      IF A$(A+1)="." THEN 8830\*LOOK FOR END OF LITERAL LIST.
8770      IF A$(A)=A$(A+1) THEN 8780\*LOOK FOR IDENTICAL LITERALS.
8771      *NOT IDENTICAL.  CONTINUE SEARCH.\GOTO 8820
8780      C=C+1                  \*INC COUNTER.
8790      FOR B0=A TO 20         \*REMOVE DUPLICATE LITERAL FROM LIST.
8800          A$(B0)=A$(B0+1)\*SHIFT A LITERAL.
8805          IF A$(B0)="." THEN 8820\*END OF LIST?
8810      NEXT B0                \*CONTINUE SHIFTING.
8820  NEXT A                     \*REPEAT FOR NEXT PAIR.
8830  Z$=""                      \*CLEAR TEMPORARY VARIABLE.
8835  FOR A=1 TO 20              \*BEGIN RECONSTRUCTION OF PRODUCT TERM.
8840      IF A$(A)="." THEN 8870\*END OF LITERAL LIST?
8850      Z$=Z$+A$(A)            \*NO.  APPEND TO TERM.
8860  NEXT A                     \*REPEAT FOR NEXT LITERAL.
```

```
8870 RETURN                    \*RETURN AND SUBROUTINE.
9000*****************************************************************
9001*          SUBROUTINE TO TRANSFER A FN FROM AN ARRAY TO A FILE.
9002*   FUNCTION IS ALWAYS TRANSFERRED FROM ARRAY H.  THE FILE IS
9003*   EITHER OLD1, OLD0, NEW0, OR NEW1 AS A FUNCTION OF J AND S.
9004*   CALLED BY THE MAIN PROGRAM (SIM2).
9005*****************************************************************
9008 INPUT=0                   \*RETURN TO NORMAL INPUT MODE.
9010 CLOSE :G                  \*CLOSE FILE ASSIGNED TO CHANNEL G.
9012 ON G GOTO 9014,9016,9018,9020\*OPEN FILE TO CHANNEL G.
9014 OPEN "OLD1" TO :1, INPUT UPDATE
9015 *GOTO NEXT STEP    \GOTO 9022
9016 OPEN "OLD0" TO :2, INPUT UPDATE
9017 *GOTO NEXT STEP    \GOTO 9022
9018 OPEN "NEW0" TO :3, INPUT UPDATE
9019 *GOTO NEXT STEP    \GOTO 9022
9020 OPEN "NEW1" TO :4, INPUT UPDATE
9022 X$=N(I1,1)                \*GET FUNCTION NAME.
9025 IF J<>0 THEN 9040 \*CHECK FUNCTION FLAG.
9030 X$=X$+"-"                 \*0-FN BEING TRANSFERED.  ADD - TO NAME.
9040 Z$=STR(T)                 \*CONVERT TIME VARIABLE TO STRING.
9041 X$=X$+Z$(:2)              \*APPEND TIME TAG TO FN NAME.
9045 X$=X$+"="                 \*APPEND = TO FUNCTION NAME.
9060 PRINT :J-2*S*J-S+3;Y5*(I1-1)+1,X$\*LOAD NAME IN FILE 1ST RCD.
9070 FOR X=1 TO 1000    \*BEGIN TRANSFER OF TERMS.
9075    INPUT :H;X,X$   \*GET TERM FROM ARRAY H.
9076    PRINT :J-2*S*J-S+3,X$\*PLACE TERM IN FILE.
9080    IF X$="." THEN 9100\*END OF FUNCTION?
9090 NEXT X                    \*NO.  REPEAT FOR NEXT TERM.
9100 INPUT=$                   \*CHANGE INPUT MODE.
9200 RETURN                    \*RETURN TO MAIN PROGRAM.
10000****************************************************************
10001*       SUBROUTINE TO COMPARE NEW FUNCTION SET TO OLD SET.
10002*  FUNCTIONS ARE ACCESSED FROM THE OLD1, OLD0, NEW0, AND NEW1
10003*  FUNCTION FILES.  IF THE TWO SETS OF FUNCTIONS ARE EQUAL,
10004*  N0=1 IS RETURNED.  OTHERWISE, N0=0 IS RETURNED.
10005*       CALLED BY MAIN PROGRAM (SIM2).
10006****************************************************************
10008 CLOSE :1\CLOSE :2\CLOSE :3\CLOSE :4\*CLOSE ALL OPEN FILES.
10010 OPEN "OLD1" TO :1, INPUT UPDATE
10011 OPEN "OLD0" TO :2, INPUT UPDATE
10012 OPEN "NEW0" TO :3, INPUT UPDATE
10013 OPEN "NEW1" TO :4, INPUT UPDATE
10015 FOR X=1 TO N1            \*BEGIN COMPARISON.
10020    INPUT :1+5*S;Y5*(X-1)+1,B$\*GET FN NAME.
10040    INPUT :4-5*S;Y5*(X-1)+1,C$\*GET FN NAME.
10045    INPUT :1+5*S,B$       \*GET TERM OF 1ST FN.
10047    INPUT :4-5*S,C$       \*GET TERM OF 2ND FN.
10050    IF B$=C$ THEN 10060   \*TERMS EQUAL?
10055    RETURN                \*NO. THEN FNS NOT EQUAL.
10060    IF B$="." THEN 10080\*YES.  END OF FN?
10070    *CONTINUE COMPARISON.\GOTO 10045
```

```
10080    INPUT :2+S;Y5*(X-1)+1,J$\*GET FN NAME.
10100    INPUT :3-S;Y5*(X-1)+1,K$\*GET FN NAME.
10105    INPUT :2+S,J$          \*GET TERM OF 1ST FN.
10107    INPUT :3-S,K$          \*GET TERM OF 2ND FN.
10110    IF J$=K$ THEN 10120 \*TERMS EQUAL?
10115    RETURN                 \*NO.  THEN FNS NOT EQUAL.
10120    IF J$="." THEN 10140\*END OF FN?
10130    *NO.  CONTINUE COMPARISON.\GOTO 10105
10140 NEXT X                    \*REPEAT FOR NEXT FUNCTIONS.
10150 N0=1                      \*FUNCTION SETS IDENTICAL.
10160 RETURN                    \*RETURN TO MAIN PROGRAM (SIM2).
11000*************************************************************************
11001*       SUBROUTINE TO PRINT FUNCTIONS ON DISPLAY DEVICE.
11002*  FUNCTIONS ARE ACCESSED FROM THE FILES CONTAINING THE
11003*  MOST CURRENT FUNCTION SET.
11004*  CURRENT INPUT-TIME AND RIPPLE-TIME IS ALSO PRINTED.
11005*************************************************************************
11010 PRINT"T=";T ,"R=";R      \*PRINT INPUT AND RIPPLE TIMES.
11020 PRINT                    \*SKIP A LINE.
11030 FOR X=1 TO N1            \*SCAN ALL ELEMENT OUTPUTS.
11040    FOR Y=3 TO 4          \*PRINT BOTH FUNCTIONS FOR EACH LINE.
11060       INPUT :Y-2*S*Y+5*S;Y5*(X-1)+1,Z$\*GET FN NAME FROM FILE.
11070       PRINT Z$;          \*PRINT THE NAME.
11075       Z=1                \*SET NAME PRINT FLAG.
11077       INPUT :Y-2*S*Y+5*S,Z$\*GET A TERM OF THE FN.
11080       IF Z$="." THEN 11305 \*END OF FN?
11090       IF Z=1 THEN 11110    \*NOT END OF FN.  CHECK NAME PRINT FLAG.
11100       Z$="+"+Z$          \*NAME NOT PRINTED LAST.  INSERT +.
11110       PRINT Z$;          \*PRINT TERM WITH LEADING +.
11120       Z=0                \*CLEAR NAME PRINT FLAG.
11130       *GET ANOTHER TERM TO PRINT.\GOTO 11077
11305       PRINT              \*SKIP TWO LINES.
11307       PRINT
11310    NEXT Y                \*REPEAT FOR NEXT FN.
11320 NEXT X                   \*REPEAT FOR NEXT LINE.
11325 INPUT=$                  \*RETURN TO COMPLETE RECORD INPUT MODE.
11330 RETURN                   \*RETURN TO MAIN PROGRAM.
13000*************************************************************************
13001*  SUBROUTINE TO DETERMINE IF A POSSIBLE OSCILLATION EXISTS.
13002*  AN OSCILLATION IS DEFINED AS THE FAILURE OF THE CIRCUIT TO
13003*  REACH A STABLE CONDITION AFTER A SPECIFIED RIPPLE-TIME
13004*  INCREMENT.  THIS INCREMENT IS EQUAL TO THE RIPPLE-TIME
13005*  LIMIT ENTERED BY THE USER AT THE BEGINNING OF THE EXECUTION
13006*  OF SIM2.  A MESSAGE IS PRINTED IF A POSSIBLE OSCILLATION
13007*  IS DETECTED.  A NEW RIPPLE-TIME LIMIT MAY THEN BE ENTERED
13008*  IF SO DESIRED.
13009*************************************************************************
13010 IF R-R1<R0 THEN 13100\*COMPARE RIPPLE-TIME INC WITH LIMIT.
13019 *PRINT MESSAGE IF LIMIT IS EQUALLED OR EXCEEDED.
13020 PRINT"POSSIBLE OSCILLATION.  ENTER NEW RIPPLE-TIME LIMIT OR <CR> TO STOP
13030 INPUT Z$                 \*ENTER RESPONSE TO QUERY.
13040 IF Z$=" " THEN 13200  \*CHECK FOR <CR> ENTRY.
```

```
13050 R0=VAL(Z$)              \*VALUE ENTERED. CONVERT TO NUMERIC.
13100 02=1                    \*SET FLAG FOR CONTINUING.
13150 RETURN                  \*RETURN TO MAIN PROGRAM.
13200 02=0                    \*CLEAR FLAG TO STOP SIMULATION.
13210 RETURN                  \*RETURN TO MAIN PROGRAM.
14000*****************************************************************
14001*          D FLIP-FLOP SIMULATION ROUTINE.
14002*****************************************************************
14010 PRINT"DFF SIMULATION ROUTINE NOT YET IMPLEMENTED."
14020 RETURN
17000*****************************************************************
17001*       SUBROUTINE TO MINIMIZE FUNCTIONS.
17002*  A FUNCTION IS SAID TO BE MINIMIZED (IN THIS CONTEXT) IF IT
17003*  CONTAINS NO REDUNDANT PRODUCT TERMS.  HENCE THE SUBROUTINE
17004*  REMOVES DUPLICATE TERMS AND TERMS COVERED BY ANOTHER TERM.
17005*       A FUNCTION IS PASSED TO THE SUBROUTINE IN ARRAY H.
17006*  THE REDUCED FUNCTION IS RETURNED IN ARRAY H.
17007*       CALLED BY THE AND SUBROUTINE AND THE OR SUBROUTINE.
17008*****************************************************************
17010 FOR X=1 TO X2-1         \*EXAMINE EACH TERM OF THE FN.
17015     INPUT :H;X,Z$       \*GET A TERM FROM ARRAY H.
17016     IF Z$="." THEN 17220\*END OF FUNCTION?
17020     X$=Z$               \*NO. COPY TERM TO X$ FOR PARSING.
17030     *PARSE TERM INTO LITERALS.\GOSUB 7800
17040     FOR Y=1 TO 20       \*COPY LITERALS TO ARRAY D$.
17050         D$(Y)=A$(Y)\*COPY A TERM.
17060         IF A$(Y)="." THEN 17080\*END OF LIST?
17070     NEXT Y              \*NO.  REPEAT FOR NEXT LITERAL.
17080     FOR Y=X+1 TO X2-1\*COMPARE ABOVE TERM TO ALL OTHER TERMS.
17090         INPUT :H;Y,Z$\*GET A TERM FROM ARRAY H.
17091         IF Z$="." THEN 17210\*END OF FN?
17100         X$=Z$           \*COPY TERM TO X$ FOR PARSING.
17110         *PARSE THE TERM INTO LITERALS\GOSUB 7800
17115         F5=1            \*INITIALIZE FLAG INDICATING
17116                         *A REDUNDANT TERM.
17120         FOR B=1 TO 20\*COMPARE LITERAL LISTS OF TWO TERMS.
17122             IF F5=0 THEN 17200\*CHECK FLAG FOR NO REDUNDANCY.
17125             IF D$(B)="." THEN 17170\*END OF LIST?
17130             FOR A=1 TO 20\*SCAN LIST OF 2ND TERM.
17135                 IF A$(A)="." THEN 17165\*END OF LIST?
17140                 IF A$(A)=D$(B) THEN 17160
17141                     *COMPARE LITERALS OF TWO LISTS.
17150                 F5=0 \*LITERALS ARE NOT ALIKE.
17151                     *CLEAR FLAG TO INDICATE NO REDUNDANCY.
17155             NEXT A      \*REPEAT FOR NEXT LITERAL OF 2ND TERM
17160             F5=1        \*MATCH FOUND. POSSIBLE REDUNDANCY.
17165         NEXT B          \*REPEAT FOR NEXT LITERAL OF 1ST TERM
17170         FOR C=Y TO X2-1\*2ND TERM REDUNDANT.  DELETE FROM H.
17180             INPUT :H;C+1,Z$\*GET A TERM FROM H.
17185             PRINT :H;C,Z$  \*PLACE TERM IN NEW POSITION.
17190         NEXT C          \*REPEAT FOR NEXT TERM.
17195         *GET NEW 2ND TERM.\GOTO 17090
```

```
17200         NEXT Y                \*REPEAT FOR NEXT TERM.
17210 NEXT X                        \*REPEAT FOR NEXT FIRST TERM.
17220 RETURN                        \*RETURN TO CALLING PROGRAM.
18000*****************************************************************
18001*        SUBROUTINE FOR CHANGING INPUT MODE.
18002*  CALLED BY MAIN ROUTINE OF SIM2.
18003*****************************************************************
18009*  PRINT MODE SELECTION MESSAGE.
18010 PRINT"SELECT DESIRED MODE."
18020 PRINT"      (1) FIXED INPUT VALUES"
18030 PRINT"      (2) INPUT VARIABLES"
18040 INPUT Y                       \*INPUT MODE SELECTION NUMBER.
18050 IF Y<3 THEN 18070             \*CHECK FOR VALID MODE NUMBER.
18055*PRINT ERROR MESSAGE.
18056 PRINT"INVALID MODE NUMBER ENTERED.  MAKE ANOTHER TRY."
18057 *REPEAT THE ENTRY.            \GOTO 18010
18060 RETURN                        \*RETURN TO MAIN PROGRAM.
18070 FOR X=1 TO 10           \*CLEAR INPUT DESCRIPTION OF ALL CONSTANTS.
18080    IF I(X,2)="FF" THEN 18115\*LOOK FOR FAULT-FREE ENTRY.
18090    IF I(X,2)="S0" THEN 18115\*LOOK FOR STUCK-AT-0 ENTRY.
18100    IF I(X,2)="S1" THEN 18115\*LOOK FOR STUCK-AT-1 ENTRY.
18110    I(X,2)="FF"          \*REPLACE CONSTANTS WITH FAULT-FREE SYMBOL.
18115 NEXT X                        \*REPEAT FOR NEXT ENTRY.
18117 IF Y=2 THEN 18300             \*CHECK FOR VARIABLE INPUT MODE.
18119 *  PRINT FIXED-INPUT ENTRY MESSAGE.
18120 PRINT"ENTER FIXED INPUT VALUES."
18130 PRINT"ENTER LINE NAME=0 OR =1 OR TO STOP ENTER /";
18140 INPUT Z$                      \*ENTER LINE NAME AND VALUE.
18150 IF Z$="/" THEN 18300          \*END OF ENTRY?
18160 FOR X=1 TO 100                \*PARSE Z$.
18170    IF Z$(X,1)="=" THEN 18190\*LOOK FOR END OF LINE NAME.
18180 NEXT X                        \*EXAMINE NEXT CHARACTER.
18190 C$=Z$(1,X-1)                  \*LET C$=LINE-NAME.
18200 B$=Z$(X+1)                    \*LET B$=LINE-VALUE.
18210 FOR X=1 TO 10                 \*SCAN I ARRAY FOR LINE-NAME ENTRIES.
18220    IF C$<>I(X,1) THEN 18240\*LOOK FOR A MATCH.
18230    I(X,2)=B$                  \*MATCH FOUND.  LOAD CONSTANT.
18240 NEXT X                        \*REPEAT FOR NEXT ENTRY.
18250 *REPEAT FOR POSSIBLE NEW ENTRY.\GOTO 18130
18300 FOR X=1 TO 10                 \*PRINT THE INPUT DESCRIPTION ARRAY.
18302    PRINT I(X,1),I(X,2),I(X,3)
18304 NEXT X
18310 RETURN                        \*RETURN TO MAIN PROGRAM.
20000 END
```

```
08:16 OCT 12,'79 DC/RACEDOC.BCARROLL

1*       SSSSS   III   M   M   LL       OOOOO   GGGGG
2*       S        I    MM MM   L       O   O   G
3*       SSSSS    I    M M M   L       O   O   G  GG
4*           S    I    M   M   L       O   O   G   G
5*       SSSSS   III   M   M   LLLLL   OOOOO   GGGGG
6*
7*            PROGRAM:        RACEDOC
8*            VERSION:        SIGMA 5
9*            REVISION:       ORIGINAL
10*           DATE:           10/6/79
11*           PROGRAMMER:     B. D. CARROLL
12*
13*****************************************************************************
14*
15*            CONTRACT SUPPORT
16*****************************************************************************
17*
18*       THIS PROGRAM WAS DEVELOPED FOR NASA MARSHALL FLIGHT
19*       CENTER UNDER CONTRACT NAS8-31572.
20*
21*****************************************************************************
22*
23*            MODIFICATION HISTORY
24*
25*****************************************************************************
26*
39*
100****************************************************************************
101*
102*            PROGRAM DESCRIPTION
103*
104****************************************************************************
105*
106*     SIMLOG IS A PROGRAM FOR SIMULATING LOGIC CIRCUITS.
107*     BOTH COMBINATIONAL AND SEQUENTIAL CIRCUITS CAN BE
108*     SIMULATED.  CIRCUITS CAN BE SIMULATED FAULT-FREE OR
109*     WITH SINGLE OR MULTIPLE STUCK TYPE FAULTS.
110*     LOGIC ELEMENTS ACCOMMODATED BY THE SIMULATOR ARE
111*     NAND GATES AND NOR GATES.  OTHER ELEMENTS ARE TO
112*     BE ADDED.
113*
114*     TWO VERSIONS OF SIMLOG HAVE BEEN WRITTEN.
115*     ONE VERSION HAS BEEN WRITTEN IN BASIC-PLUS FOR
116*     EXECUTION ON A PDP11/40 RSTS/E SYSTEM.
117*     ANOTHER VERSION HAS BEEN WRITTEN IN XEROX BASIC
118*     FOR EXECUTION ON A SIGMA 5 CPV SYSTEM.
119*
120*     THE PDP11/40 VERSION IS PARTITIONED INTO TWO
121*     SUBPROGRAMS, SIMA AND SIMB.  THIS PARTITIONING WAS
122*     NECESSARY DUE TO THE LARGE MEMORY REQUIREMENTS OF
```

```
123*    THE PROGRAM.  VIRTUAL ARRAYS ARE USED FOR ALL LARGE
124*    ARRAY STORAGE.
125*
126*    THE SIGMA 5 VERSION IS PARTITIONED INTO THREE
127*    SUBPROGRAMS, SIM1, SIM2, 1ND RACE.  THIS DEGREE OF
128*    PARTITIONING WAS NECESSARY SINCE VIRTUAL ARRAYS ARE
129*    NOT AVAILABLE ON THE SIGMA 5.
130*
131*    THE FUNCTION OF SIM1 IS TO INPUT ALL DATA NEEDED
132*    CONCERNING CIRCUIT DESCRIPTION, FAULT DESCRIPTION,
133*    INITIAL CONDITIONS, AND SIMULATION MODES.
134*
135*    THE FUNCTION OF SIM2 IS TO PERFORM THE ACTUAL
136*    SIMULATION COMPUTATIONS WITH THE EXCEPTION OF
137*    RACE ANALYSIS AND TO OUTPUT THE SIMULATION RESULTS.
138*
139*    THE FUNCTION OF RACE IS TO PERFORM THE RACE
140*    ANALYSIS COMPUTATIONS.
141*
142*
300*****************************************************************
301*
302*         I/O STREAMS
303*
304*****************************************************************
305*
306*
309*    STREAM          USAGE
310*
311*
312*       1         FILE OF OLD ONE-FUNCTIONS(S=0)
313*                 FILE OF NEW ONE-FUNCTIONS(S=1)
314*
315*       2         FILE OF OLD ZERO-FUNCTIONS(S=0)
316*                 FILE OF NEW ZERO-FUNCTIONS(S=1)
317*
318*       3         FILE OF NEW ZERO-FUNCTIONS(S=0)
319*                 FILE OF OLD ZERO-FUNCTIONS(S=1)
320*
321*       4         FILE OF NEW ONE-FUNCTIONS(S=0)
322*                 FILE OF OLD ONE-FUNCTIONS(S=1)
323*
324*
325*       THE ABOVE REPRESENTS THE BASIC ASSIGNMENTS OF I/O STREAMS
326*  TO FUNCTION FILES.  HOWEVER, THE SIGMA 5 PROVIDES ONLY 4 I/O
327*  STREAM NUMBERS.  HENCE, IT BECOMES NECESSARY TO REASSIGN STREAMS
328*  WHEN ACCESS TO ARRAY-FILES IS NEEDED.  THIS REASSIGNMENT IS
329*  DYNAMIC AS A FUNCTION OF PARAMETERS S AND X7 SINCE IT IS
330*  NECESSARY TO KEEP ACCESS TO ONE FUNCTION FILE ALSO.
331*  THE FOLLOWING TABLE DETAILS THIS DYNAMIC ASSIGNMENT.
332*
333*-------|-------------------------|-----------------------------|
```

```
334*        !         S=0                    !              S=1                    !
335*--------!--------------------------------!-------------------------------------!
336*STREAM#! X7=0       !      X7=1          !   X7=0       !      X7=1            !
337*--------!------------!--------------------!--------------!--------------------!
338*   1    ! GARRAY     ! FARRAY             ! HARRAY       !    OLD1             !
339*--------!------------!--------------------!--------------!--------------------!
340*   2    ! FARRAY     ! GARRAY             !    OLD0      ! HARRAY             !
341*--------!------------!--------------------!--------------!--------------------!
342*   3    !    NEW0    ! HARRAY             ! FARRAY       ! GARRAY             !
343*--------!------------!--------------------!--------------!--------------------!
344*   4    ! HARRAY     !    NEW1            ! GARRAY       ! FARRAY             !
345*--------!------------!--------------------!--------------!--------------------!
346*
400*****************************************************************************
401*
402*         VARIABLE DEFINITIONS
403*     (MAIN PROGRAM ONLY.  SEE SIM2DOC FOR SUBROUTINE VARIABLES.)
404*
405*****************************************************************************
406*
409*    VARIABLE        DEFINITION
410*
423 *         B$  USED AS A TEMPORARY STRING VARIABLE.
424 *
425 *         E$,K$,S$,V$,J$,L$
426 *             USED AS TEMPORARY STRING VARIABLES.
427 *
440 *         F       INDICATES THE I/O STREAM TO WHICH F ARRAY FILE
441 *               IS OPENED.
442 *
455 *         G       INDICATES THE I/O STREAM TO WHICH THE G ARRAY FILE
456 *               IS OPENED.
457 *
460 *         H       INDICATES THE I/O STREAM TO WHICH THE H ARRAY FILE
461 *               IS OPENED.
462 *
463 *        H9       INDICATES THE UPPER LIMIT OF THE LOOP VARIABLE
464 *               OF SOME FOR-NEXT LOOPS.  SHOULD BE SET >= THE
465 *               NUMBER OF ROWS IN THE I ARRAY.
466 *
515 *         N1      INDICATES THE NUMBER OF ELEMENTS IN THE CIRCUIT.
516 *               CONSEQUENTLY THE NUMBER OF ROWS IN N AND P.
517 *
540 *         Q0      CONTROL VARIABLE IN A FOR-NEXT LOOP.
541 *
542 *         Q1      THE NUMBER OF CROSS-COUPLED PAIRS OF ELEMENTS IN
543 *               THE CIRCUIT.  ALSO THE NUMBER OF ROWS IN Q.
544 *
557 *         S       A SWITCH INDICATING WHICH CHANNEL TO USE WHEN
558 *               ACCESSING FUNCTION FILES.
559 *
567 *         V       A TEMPORARY VARIABLE USED TO REPRESENT THE
```

```
568 *              NUMERIC VALUE OF A STRING VARIABLE.
569 *
573 *         X    USED AS THE CONTROL VARIABLE IN FOR-NEXT LOOPS.
574 *
597 *         Y5   A PARAMETER INDICATING THE MAXIMUM NUMBER OF TERMS
598 *              THAT CAN BE STORED IN THE FUNCTION FILE OF EACH
599 *              FUNCTION.
600 *
800*****************************************************************
801*
802*          SUBROUTINE DESCRIPTIONS
803*
804*****************************************************************
805*
809*          PROGRAMMER DEFINED SUBROUTINES
810*
811*     LINE            DESCRIPTION
812*
813*
824*     7200        TERM SORTING.
825*
826*     7800        PRODUCT TERM DISASSEMBLY INTO LITERAL COMPONENTS.
827*
828*     8000        AND SUBROUTINE.
829*
830*     8250        ZERO PRODUCT TERM DETECTION.
831*
832*     8650        LITERAL SORTING.
833*
842*     17000       FUNCTION MINIMIZATION.
843*
900*****************************************************************
901*
902*          DIMENSION STATEMENTS
903*
904*****************************************************************
905*
910       DIM N(200,4), I(400,3), Q(50,2), P(200,2)
920       DIM A$(51), D$(51)
930       DIM U(1)
999*
1000****************************************************************
1001****************************************************************
1002*                                                              *
1003*           START OF MAIN PROGRAM                              *
1004*                                                              *
1005****************************************************************
1006****************************************************************
1007*
1008*
1010 OPEN "OLD1" TO :1, INPUT UPDATE
1020 OPEN "OLD0" TO :2, INPUT UPDATE
```

```
1030 OPEN "NEW0" TO :3, INPUT UPDATE
1040 OPEN "NEW1" TO :4, INPUT UPDATE
1155 INPUT=0
1160 FOR Q0=1 TO Q1     \*BEGIN PROCESSING OF EACH XC PAIR.
1165    FOR X=1 TO N1   \*SEARCH FOR XC ELEMENT IN ELEMENT ARRAY.
1170         IF N(X,1)=Q(Q0,1) THEN 1180
1175    NEXT X
1179* CHECK ELEMENT TYPE.
1180    IF N(X,2)="NAND" THEN 1190
1185    IF N(X,2)="NOR" THEN 1200
1190    X7=0            \*LOAD FLAG FOR NAND GATE.
1195    GOTO 1205
1200    X7=1            \*LOAD FLAG FOR NOR GATE.
1205    V=VAL(Q(Q0,1)) \*CONVERT ELEMENT OFFSET TO NUMERIC.
1209 *       GET NAME LABELS OF FIRST TWO ONE-FN'S.
1210    INPUT :3+X7-S-2*X7*S;Y5*(V-1)+1,B$
1215    INPUT :2-X7+S+2*X7*S;Y5*(V-1)+1,E$
1219 *       GET TERM OF FIRST TWO ONE-FN'S.
1220    INPUT :3+X7-S-2*X7*S,B$
1225    INPUT :2-X7+S+2*X7*S,E$
1229 *  CHECK FOR EQUALITY OF FUNCTIONS.
1230    IF B$<>E$ THEN 1245
1235    IF B$="." THEN 1285 \*IF YES, FN'S ARE EQUAL.
1240    *IF NO, THEN EXAMINE NEXT TERMS.\GOTO 1220
1245    V=VAL(Q(Q0,2)) \*CONVERT ELEMENT INDEX TO NUMERIC.
1249 *  GET NAME LABELS OF FIRST TWO ZERO-FN'S.
1250    INPUT :4-X7-3*S+2*X7*S;Y5*(V-1)+1,K$
1255    INPUT :1+X7+3*S-2*X7*S;Y5*(V-1)+1,S$
1259 *  GET TERMS OF FIRST TWO ZERO-FN'S.
1260    INPUT :4-X7-3*S+2*X7*S,K$
1265    INPUT :1+X7+3*S-2*X7*S,S$
1269 *  CHECK FOR FUNCTION EQUALITY.
1270    IF K$<>S$ THEN 1365
1275    IF K$="." THEN 1285 \*IF YES, THEN FN'S ARE EQUAL.
1280    *IF NO, EXAMINE NEXT TERMS.   \GOTO 1260
1285    V=VAL(Q(Q0,2)) \*CONVERT ELEMENT INDEX TO NUMERIC.
1289 *  GET NAME LABELS OF SECOND TWO ZERO-FN'S.
1290    INPUT :3+X7-S-2*X7*S;Y5*(V-1)+1,V$
1295    INPUT :2-X7+S+2*X7*S;Y5*(V-1)+1,Z$
1299 *  GET TERM OF SECOND TWO ZERO-FN'S.
1300    INPUT :3+X7-S-2*X7*S,V$
1305    INPUT :2-X7+S+2*X7*S,Z$
1310    IF V$<>Z$ THEN 1325 \*CHECK FOR EQUALITY OF FUNCTIONS.
1315    IF V$="." THEN 1830 \*IF YES, THEN FN'S ARE EQUAL.
1320    *IF NO, EXAMINE NEXT TERMS.\GOTO 1300
1325    V=VAL(Q(Q0,1)) \*CONVERT ELEMENT INDEX TO NUMERIC.
1329    * GET NAME LABELS OF SECOND TWO ONE-FUNCTIONS.
1330    INPUT :4-X7-3*S+2*X7*S;Y5*(V-1)+1,J$
1335    INPUT :1+X7+3*S-2*X7*S;Y5*(V-1)+1,L$
1339 *  GET TERMS OF SECOND TWO ONE-FUNCTIONS.
1340    INPUT :4-X7-3*S+2*X7*S,J$
1345    INPUT :1+X7+3*S-2*X7*S,L$
```

```
1350    IF J$<>L$ THEN 1365 \*CHECK FOR EQUALITY OF FUNCTIONS.
1355    IF J$="." THEN 1830 \*IF TRUE, FUNCTIONS ARE EQUAL.
1360    *IF NO, EXAMINE NEXT TWO TERMS.\GOTO 1340
1364    *CLOSE FUNCTION FILES F AND G.
1365    CLOSE :2-X7+S+2*X7*S
1370    CLOSE :1+X7+3*S-2*X7*S
1375    F=2-X7+S+2*X7*S
1380    G=1+X7+3*S-2*X7*S
1384    *OPEN FUNCTION FILES FARRAY AND GARRAY.
1385    OPEN "FARRAY" TO :F,INPUT UPDATE
1390    OPEN "GARRAY" TO :G,INPUT UPDATE
1394    *CHECK FIRST ZERO-FN FOR ZERO-VALUE.
1395    V=VAL(Q(Q0,1))
1400    INPUT :3+X7-S-2*X7*S;Y5*(V-1)+2,B$
1405    IF B$="0" THEN 1645
1409    *CHECK FIRST NEW ONE-FN FOR ZERO-VALUE.
1410    V=VAL(Q(Q0,2))
1415    INPUT :4-X7-3*S+2*X7*S;Y5*(V-1)+2,K$
1420    IF K$="0" THEN 1550
1424    *LOAD FIRST NEW ZERO-FN INTO FARRAY.
1425    PRINT :F;1,B$
1430    FOR X=2 TO H9
1435         INPUT :3+X7-S-2*X7*S,B$
1440         PRINT :F;X,B$
1445         IF B$="." THEN 1455
1450    NEXT X
1454    *LOAD FIRST NEW ONE-FN IN GARRAY.
1455    PRINT :G;1,K$
1460    FOR X=2 TO H9
1465         INPUT :4-X7-3*S+2*X7*S,K$
1470         PRINT :G;X,K$
1475         IF K$="." THEN 1485
1480    NEXT X
1484    *CLOSE FUNCTION FILE H.
1485    CLOSE :4-X7-3*S+2*X7*S
1490    H=4-X7-3*S+2*X7*S
1494    *OPEN ARRAY FILE HARRAY TO H.
1495    OPEN "HARRAY" TO :H,INPUT UPDATE
1499    *AND FIRST 0-FN AND FIRST 1-FN.
1500    GOSUB 8000
1504    *REPLACE FIRST NEW ZERO-FN.
1505    V=VAL(Q(Q0,1))
1510    INPUT :H;1,Z$
1515    PRINT :3+X7-S-2*X7*S;Y5*(V-1)+2,Z$
1520    FOR X=2 TO H9
1525         INPUT :H;X,Z$
1530         PRINT :3+X7-S-2*X7*S,Z$
1535         IF Z$="." THEN 1590
1540    NEXT X
1545    GOTO 1590
1549    *REPLACE FIRST NEW ONE-FN.
1550    V=VAL(Q(Q0,2))
```

```
1555    PRINT :4-X7-3*S+2*X7*S;Y5*(V-1)+2,B$
1560    FOR X=1 TO H9
1565         INPUT :3+X7-S-2*X7*S,B$
1570         PRINT :4-X7-3*S+2*X7*S,B$
1575         IF B$="." THEN 1645
1580    NEXT X
1585    GOTO 1645
1589    *CLOSE ARRAY FILE H.
1590    CLOSE :4-X7-3*S+2*X7*S
1594    *OPEN APPROPRIATE FUNCTION FILE TO H.
1595    IF S=1 THEN 1625
1600    IF X7=1 THEN 1615
1605    OPEN "NEW1" TO :4,INPUT UPDATE
1610    GOTO 1645
1615    OPEN "NEW0" TO :3,INPUT UPDATE
1620    GOTO 1645
1625    IF X7=1 THEN 1640
1630    OPEN "OLD1" TO :1,INPUT UPDATE
1635    GOTO 1645
1640    OPEN "OLD0" TO :2,INPUT UPDATE
1644    *CHECK SECOND NEW 0-FN FOR 0-VALUE.
1645    V=VAL(Q(Q0,2))
1650    INPUT :3+X7-S-2*X7*S;Y5*(V-1)+2,V$
1655    IF V$="0" THEN 1830
1660    V=VAL(Q(Q0,1))
1665    INPUT :4-X7-3*S+2*X7*S;Y5*(V-1)+2,J$
1670    IF J$="0" THEN 1795
1674    *LOAD SECOND NEW 0-FN IN FARRAY.
1675    PRINT :F;1,V$
1680    FOR X=2 TO H9
1685         INPUT :3+X7-S-2*X7*S,V$
1690         PRINT :F;X,V$
1695         IF V$="." THEN 1705
1700    NEXT X
1704    *LOAD SECOND NEW 1-FN IN GARRAY.
1705    PRINT :G;1,J$
1710    FOR X=2 TO H9
1715         INPUT :4-X7-3*S+2*X7*S,J$
1720         PRINT :G;X,J$
1725         IF J$="." THEN 1735
1730    NEXT X
1734    *CLOSE FUNCTION FILE H.
1735    CLOSE :4-X7-3*S+2*X7*S
1739    *OPEN ARRAY-FILE HARRAY TO H.
1740    OPEN "HARRAY" TO :H,INPUT UPDATE
1744    *AND 2ND 0-FN WITH 2ND 1-FN.
1745    GOSUB 8000
1749    *REPLACE SECOND 0-FN.
1750    V=VAL(Q(Q0,2))
1755    INPUT :H;1,Z$
1760    PRINT :3+X7-S-2*X7*S;Y5*(V-1)+2,Z$
1765    FOR X=2 TO H9
```

```
1770          INPUT :H;X,Z$
1775          PRINT :3+X7-S-2*X7*S,Z$
1780          IF Z$="." THEN 1830
1785     NEXT X
1790     GOTO 1830
1794     *REPLACE SECOND 1-FN.
1795     V=VAL(Q(Q0,1))
1800     PRINT :4-X7-3*S+2*X7*S;Y5*(V-1)+2,V$
1805     FOR X=1 TO H9
1810          INPUT :3+X7-S-2*X7*S,V$
1815          PRINT :4-X7-3*S+2*X7*S,V$
1820          IF V$="." THEN 1830
1825     NEXT X
1829     *CLOSE ALL FILES.
1830     CLOSE :1
1835     CLOSE :2
1840     CLOSE :3
1845     CLOSE :4
1849     *REOPEN FUNCTION FILES TO INITIAL ASSIGNMENT.
1850     OPEN "OLD1" TO :1,INPUT UPDATE
1855     OPEN "OLD0" TO :2,INPUT UPDATE
1860     OPEN "NEW0" TO :3,INPUT UPDATE
1865     OPEN "NEW1" TO :4,INPUT UPDATE
1870 NEXT Q0       \*END OF PRIMARY LOOP.
1875 INPUT=$
1879 *CLOSE ALL FILES BEFORE CHAINING.
1880 CLOSE :1
1885 CLOSE :2
1890 CLOSE :3
1895 CLOSE :4
1900 C0=2
1905 CHAIN LINK "SIM2DOC"
7200*****************************************************************
7201*          SUBROUTINE TO SORT THE TERMS OF A FUNCTION.
7202*          FUNCTION MUST BE STORED IN ARRAY H BEFORE
7203*          SORTING RULES: LENGTH OF TERM, ALPHABETICAL, NUMERICAL.
7204*          CALLED BY OR AND AND SUBROUTINES.
7205*          CALLS NO SUBROUTINES.
7206*****************************************************************
7220 FOR X=1 TO 100          \*COUNT NUMBER OF TERMS IN FN.
7230     INPUT :H;X,Z$        \*GET A TERM.
7231     IF Z$="." THEN 7250 \*END OF FUNCTION?
7240 NEXT X                  \*NO.
7249 *X = THE NUMBER OF TERMS IN H PLUS 1.
7250 IF X=2 THEN 7500        \*CHECK FOR SINGLE TERM FN.
7252 X0=X                    \*SAVE X FOR USE AS SORT LOOP LIMITER.
7255 X1=X                    \*SAVE X FOR STORE LOOP LIMITER.
7260 FOR X=1 TO X0-2    \*USE A BUBBLE SORT TO ORDER THE TERMS.
7270     INPUT :H;X,X$   \*GET FIRST TERM OF H.
7272     INPUT :H;X+1,Y$ \*GET ADJACENT TERM OF H.
7274     IF X$<Y$ THEN 7310\*COMPARE THE TERMS.
7280     *REVERSE THE ORDER OF THE TWO TERMS.
```

```
7290     PRINT :H;X,Y$
7300     PRINT :H;X+1,X$
7310 NEXT X              \*REPEAT FOR NEXT PAIR OF TERMS.
7319 * ONE TERM HAS BEEN PLACED.IN ITS PROPER PLACE.
7320 X0=X0-1             \*DECREMENT SORT LOOP LIMITER.
7330 IF X0>2 THEN 7260 \*HAVE ALL TERMS BEEN ORDERED?
7340 X0=X1               \*YES.  RESTORE X0.
7344 *DUPLICATE TERMS WILL NOW BE REMOVED.
7345 FOR X=1 TO X1       \*COMPARE ADJACENT TERMS FOR EQUALITY.
7347     INPUT :H;X,X$   \*GET A TERM.
7348     IF X$="." THEN 7420\*END OF FUNCTION?
7350     INPUT :H;X+1,Y$\*NO.  GET ADJACENT TERM.
7351     IF X$=Y$ THEN 7360\*ARE TERMS IDENTICAL?
7355     *NO.  GET NEXT PAIR.\GOTO 7410
7360     X0=X0-1         \*YES.  DECREMENT TERM COUNT.
7365     FOR Y=X+1 TO X1-1\*REMOVE DUPLICATE TERM AND COMPACT H.
7370          INPUT :H;Y+1,X$\*GET TERM FROM H.
7371          PRINT :H;Y,X$\*MOVE TERM TO NEW POSITION IN H.
7380          IF X$="." THEN 7410\*END OF FUNCTION?
7390     NEXT Y          \*REPEAT FOR NEXT TERM.
7410 NEXT X              \*REPEAT FOR NEXT PAIR OF TERMS.
7420 FOR X=1 TO X0-2     \*SORT TERMS BY LENGTH USING BUBBLE SORT.
7430     INPUT :H;X,X$
7431     INPUT :H;X+1,Y$
7432     L1=LEN(X$)
7433     L2=LEN(Y$)
7434     IF L1<=L2 THEN 7470\*IF TRUE, DO NOT REORDER.
7440     PRINT :H;X,Y$   \*REORDER.
7450     PRINT :H;X+1,X$
7470 NEXT X              \*REPEAT FOR NEXT PAIR.
7480 X0=X0-1             \*DECREMENT LOOP LIMIT FOR NEXT PASS.
7490 IF X0>2 THEN 7420 \*SORTING COMPLETE?
7500 RETURN              \*YES.  RETURN TO CALLING PROGRAM.
7800*****************************************************************
7801*         SUBROUTINE TO DISASSEMBLE A PRODUCT TERM INTO ITS
7802*    LITERAL COMPONENTS.  A LITERAL IS A VARIABLE NAME OR A
7803*    COMPLEMENTED VARIABLE NAME TOGETHER WITH ITS TIME TAG
7804*    OR FAULT SYMBOL.
7805*    TERM IS PASSED TO SUBROUTINE IN X$.  LITERALS RETURNED
7806*    IN A$ ARRAY.  CALLED BY LITERAL SORT SUBROUTINE.
7807*****************************************************************
7810 F5=0                \*CLEAR SPECIAL CHARACTER FLAG.
7815 Z$=""               \*CLEAR DUMMY STRING VARIABLE.
7820 A=1                 \*INITIALIZE POINTER TO A$ ARRAY.
7830 L=LEN(X$)           \*COMPUTE LENGTH OF TERM.
7835 FOR X3=1 TO L       \*BEGIN DISASSEMBLY OF TERM.
7840     IF X$(:X3,1)="-" THEN 7900\*LOOK FOR SPECIAL CHARS (-).
7842     IF X$(:X3,1)>"Z" THEN 7900\*LOOK FOR TIME TAG (NUMERAL).
7844     IF X$(:X3,1)<"A" THEN 7900\*LOOK FOR FAULT SYMBOLS (1/*).
7850     IF F5<>1 THEN 7910\*END OF LITERAL REACHED?
7860     A$(A)=Z$        \*YES.  PLACE LITERAL IN A$ ARRAY.
7861                      *END OF LITERAL IS INDICATED WHEN A SPECIAL
```

```
7862                     *CHARACTER IS FOLLOWED BY A LETTER.
7863                     *THE LETTER IS THE FIRST CHARACTER OF THE
7864                     *NEXT LITERAL.
7870    Z$=X$(:X3,1)    \*BEGIN CONSTRUCTING NEXT LITERAL.
7880    F5=0            \*CLEAR SPECIAL CHARACTER FLAG.
7890    A=A+1           \*INC POINTER TO A$ ARRAY.
7895    GOTO 7920
7896                     *CONTINUE PROCESSING TERM.
7900    F5=1            \*SET SPECIAL CHARACTER FLAG.
7910    Z$=Z$+X$(:X3,1)\*CONSTRUCT LITERAL.
7920 NEXT X3            \*REPEAT FOR NEXT CHARACTER OF TERM.
7930 A$(A)=Z$           \*LOAD LAST LITERAL OF TERM IN A$ ARRAY.
7940 A$(A+1)="."        \*LOAD END-OF-LITERAL MARK.
7950 RETURN             \*RETURN TO LITERAL SORT SUBROUTINE.
8000*************************************************************************
8001*        SUBROUTINE TO PERFORM THE LOGICAL AND OF
8002*        TWO FUNCTIONS.
8003*        THE INPUT FUNCTIONS ARE PASSED IN ARRAYS F AND G.
8004*   THE RESULTANT FUNCTION IS RETURNED IN ARRAY H.
8005*        CALLED BY MAIN PROGRAM (SIM2) AND RACE PROGRAM.
8006*        CALLS ZERO PRODUCT DETECT SUBR (8250), LITERAL SORT
8007*   SUBR(8650), TERM SORT SUBR(7200), AND MINIMIZATION SUBR(17000)
8008*************************************************************************
8010 INPUT :F;1,X$              \*CHECK FOR ZERO FN VALUE.
8011 IF X$="0" THEN 8030
8020 INPUT :G;1,Y$              \*CHECK FOR ZERO FN VALUE.
8021 IF Y$<>"0" THEN 8050
8030 PRINT :H;1,"0"             \*RESULT IS ZERO.
8040 PRINT :H;2,"."
8045 RETURN                     \*RETURN TO CALLING PROGRAM.
8050 IF X$="1" THEN 8058        \*CHECK FOR ONE VALUED FUNCTION.
8051 IF Y$<>"1" THEN 8068       \*CHECK FOR ONE VALUED FUNCTION.
8052 FOR X=1 TO H9              \*2ND FN ONLY IS ONE VALUED.  RESULT = F.
8053    INPUT :F;X,X$           \*GET TERM OF F.
8054    PRINT :H;X,X$           \*PLACE IN H ARRAY.
8055    IF X$="." THEN 8160     \*END OF FUNCTION?
8056 NEXT X                     \*REPEAT FOR NEXT TERM.
8057 *PROCEED TO FUNCTION REDUCTION STEP.\GOTO 8160
8058 IF Y$<>"1" THEN 8062\*1-ST FN IS 1-VALUED. IS THE 2ND FN 1-VALUED?
8059 PRINT :H;1,"1"             \*BOTH FNS ARE 1-VALUED.  RESULT IS 1-VALUED.
8060 PRINT :H;2,"."
8061 RETURN                     \*RETURN TO CALLING PROGRAM.
8062 FOR X=1 TO H9              \*1ST FN IS 1-VALUED.  2ND IS NOT.
8063    INPUT :G;X,Y$           \*RESULT = G.
8064    PRINT :H;X,Y$
8065    IF Y$="." THEN 8160\*END OF FUNCTION?
8066 NEXT X
8067 *PROCEED TO MINIMIZATION STEP.\GOTO 8160
8068 W=1       \*NEITHER INPUT IS CONSTANT-VALUED. INIT POINTER TO H.
8069 PRINT :H;1,"0"             \*INITIALIZE H TO 0-VALUE.
8070 PRINT :H;2,"."
8075 FOR X=1 TO H9              \*BEGIN PRODUCT-TERM FORMATION LOOP.
```

```
8080     INPUT :F;X,X$  \*GET TERM FROM F.
8090     FOR Y=1 TO H9  \*BEGIN LOOP FOR ACCESS TO G.
8091          INPUT :G;Y,Y$\*GET TERM FROM G.
8092          Z$=X$+Y$  \*FORM PRODUCT OF TERMS.
8093          PRINT :H;W,Z$\*PLACE RESULT IN H.
8100          *CHECK FOR ZERO PRODUCT.\GOSUB 8250
8110          INPUT :G;Y+1,Y$\*CHECK G FOR END OF FUNCTION.
8111          IF Y$="." THEN 8130
8120     NEXT Y          \*NOT END OF G.  GET NEXT TERM.
8130     INPUT :F;X+1,X$\*CHECK F FOR END OF FUNCTION.
8131     IF X$="." THEN 8150
8140 NEXT X             \*NOT END OF F.  GET NEXT TERM.
8150 IF W<>1 THEN 9155 \*END OF BOTH FNS REACHED.  0-VALUE RESULT?
8152 PRINT :H;1,"0"     \*YES.  LOAD 0-VALUE IN H.
8153 RETURN             \*RETURN TO CALLING PROGRAM.
8155 PRINT :H;W,"."     \*NO.  LOAD END-OF-FUNCTION MARK.
8160 FOR X=1 TO H9      \*BEGIN MIN STEP BY ORDERING LITS IN EACH TERM.
8165     INPUT :H;X,X$  \*GET A TERM FROM H.
8167     IF X$="." THEN 8190\*END OF FUNCTION?
8170     *NO.  ORDER LITERALS.\GOSUB 8650
8175     PRINT :H;X,Z$  \*REPLACE ORDERED TERM.
8180 NEXT X             \*REPEAT FOR NEXT TERM.
8190 *ORDER TERMS OF THE FUNCTION.\GOSUB 7200
8200 *REMOVE REDUNDANT TERMS.\GOSUB 17000
8210 RETURN             \*RETURN TO CALLING PROGRAM.
8250*************************************************************************
8251*         SUBROUTINE TO CHECK FOR ZERO PRODUCT TERM.
8252*         TERM TO BE CHECKED WAS FORMED BY THE AND SUBROUTINE
8253*    AND WAS PLACED IN THE H ARRAY.  THE TERM IS RETRIEVED FROM H
8254*    FOR PROCESSING.  IF THE TERM IS ZERO, THE POINTER (W) TO THE
8255*    H ARRAY IS NOT INCREMENTED WHICH RESULTS IN THE TERM BEING
8256*    OVERWRITTEN BY ANOTHER TERM. IF THE TERM IS NON-0, W IS INC.
8257*         CALLED BY THE AND SUBROUTINE.
8258*************************************************************************
8260 C=1                \*SET THE SPECIAL CHARACTER FLAG.
8270 D=1                \*SET THE MINUS FLAG.
8280 Z$=""              \*CLEAR A TEMPORARY VARIABLE.
8290 B$=""              \*CLEAR ANOTHER TEMPORARY VARIABLE.
8300 INPUT :H;W,Y$      \*GET TERM TO BE ANALYZED.
8301 L=LEN(Y$)          \*COMPUTE  LENGTH OF TERM.
8310 FOR A=1 TO L       \*BEGIN TERM ANALYSIS.
8320     X$=Y$(:A,1)    \*EXAMINE A CHARACTER FROM Y$.
8330     IF X$>"Z" THEN 8350\*NUMERAL?
8332     IF X$="-" THEN 8350\*MINUS SIGN?
8334     IF X$="*" THEN 8350\*STUCK-AT-0 SYMBOL?
8336     IF X$="!" THEN 8350\*STUCK-AT-1 SYMBOL?
8340     IF C=1 THEN 8400\*END OF LITERAL?
8341     *YES.  LOOK FOR ITS COMPLEMENT.\GOTO 8420
8350     C=0            \*SPECIAL SYMBOL FOUND.  CLEAR FLAG.
8360     IF X$<>"-" THEN 8370\*WAS THE SYMBOL A - ?
8365     D=0            \*YES.  CLEAR FLAG.
8366     *GET NEXT CHARACTER.\GOTO 8600
```

```
8370     IF D<>1 THEN 8400\*CHECK FOR COMPLEMENTED LITERAL.
8390     Z$=Z$+"-"       \*NOT COMPLEMENTED.  INSERT MINUS.
8400     Z$=Z$+X$        \*CONSTRUCT TEST LITERAL.
8410     *GET NEXT CHARACTER.\GOTO 8600
8420     C=1                     \*SET SPECIAL CHARACTER FLAG.
8430     B1=A                    \*SAVE POINTER TO Y$.
8440     FOR B=B1 TO L           \*FIND NEXT LITERAL IN TERM.
8450          INPUT :H;W,Y$      \*GET THE TERM FROM THE FILE.
8451          X$=Y$(:B,1)        \*GET A CHARACTER OF THE TERM.
8460          IF X$>"Z" THEN 8480\*A NUMERAL?
8462          IF X$="-" THEN 8480\*A MINUS SIGN?
8464          IF X$="*" THEN 8480\*STUCK-AT-0 SYMBOL?
8466          IF X$="!" THEN 8480\*STUCK-AT-1 SYMBOL?
8470          IF C=1 THEN 8490\*END OF LITERAL?
8471          *YES.   COMPARE TO TEST LITERAL.\GOTO 8510
8480          C=0               \*SPECIAL CHARACTER. CLEAR FLAG.
8490          B$=B$+X$          \*CONSTRUCT LITERAL.
8500          *GET NEXT CHARACTER.\*GOTO 8540
8510          IF Z$=B$ THEN 8620\*COMPARE WITH TEST LITERAL.
8520          C=1               \*NOT = .  SET SPECIAL CHAR FLAG.
8530          B$=""             \*CLEAR TEMPORARY VARIABLE.
8535          *SEARCH FOR NEXT LITERAL.\GOTO 8450
8540     NEXT B                 \*REPEAT FOR NEXT CHARACTER.
8550     IF Z$=B$ THEN 8620     \*COMPARE LAST LITERAL WITH TEST.
8560     C=1                    \*SET SPECIAL CHARACTER FLAG.
8570     D=1                    \*SET THE MINUS SIGN FLAG.
8580     Z$=""                  \*CLEAR A TEMPORARY VARIABLE.
8590     B$=""                  \*CLEAR ANOTHER TEMP VARIABLE.
8595     *DEVELOP A NEW TEST LITERAL.\GOTO 8320
8600 NEXT A                     \*REPEAT FOR NEXT CHARACTER.
8610 W=W+1                      \*NON-0 TERM.  INC POINTER TO H.
8620 RETURN                     \*RETURN TO AND SUBROUTINE.
8650*****************************************************************
8651*         SUBROUTINE TO SORT THE LITERALS IN A PRODUCT TERM.
8652*    LITERALS ARE PASSED TO SUBROUTINE IN Z$ AND THE SORTED
8653*    TERM IS RETURNED IN Z$.
8654*         CALLED BY THE AND SUBROUTINE.
8655*         CALLS THE TERM DISASSEMBLY SUBROUTINE.
8656*****************************************************************
8660 *DISASSEMBLE THE TERM INTO LITERALS.\GOSUB 7800
8663 C=0                  \*INITIALIZE COUNTER.
8668 B1=A                 \*GET NUMBER OF LITERALS IN TERM FROM SUBR 7800
8670 FOR A=1 TO B1        \*SORT THE LITERALS IN A$ USING A BUBBLE SORT.
8680     IF A$(A+1)="." THEN 8740\*END OF LITERALS.
8690     IF A$(A)<A$(A+1) THEN 8730\*COMPARE TWO LITERALS.
8700     Z$=A$(A)          \*RE-ORDERING NEEDED.  SAVE ONE LITERAL.
8710     A$(A)=A$(A+1)     \*SWAP PLACES OF THE TWO.
8720     A$(A+1)=Z$        \*COMPLETE THE SWAP.
8730 NEXT A                \*REPEAT FOR NEXT PAIR.
8740 IF B1=1 THEN 8750     \*CHECK FOR END OF SORT.
8745 B1=B1-1               \*DECREASE LOOP LIMIT.  CONTINUE SORT.
8746 *REPEAT FOR NEXT PASS THROUGH THE LIST.\GOTO 8670
```

```
8750 FOR A=1 TO 20      \*SORT COMPLETE.  LOOK FOR DUPLICATE LITERALS.
8760     IF A$(A+1)="." THEN 8830\*LOOK FOR END OF LITERAL LIST.
8770     IF A$(A)=A$(A+1) THEN 8780\*LOOK FOR IDENTICAL LITERALS.
8771     *NOT IDENTICAL.  CONTINUE SEARCH.\GOTO 8820
8780     C=C+1           \*INC COUNTER.
8790     FOR B0=A TO 20 \*REMOVE DUPLICATE LITERAL FROM LIST.
8800          A$(B0)=A$(B0+1)\*SHIFT A LITERAL.
8805          IF A$(B0)="." THEN 8820\*END OF LIST?
8810     NEXT B0         \*CONTINUE SHIFTING.
8820 NEXT A              \*REPEAT FOR NEXT PAIR.
8830 Z$=""               \*CLEAR TEMPORARY VARIABLE.
8835 FOR A=1 TO 20       \*BEGIN RECONSTRUCTION OF PRODUCT TERM.
8840     IF A$(A)="." THEN 8870\*END OF LITERAL LIST?
8850     Z$=Z$+A$(A)     \*NO.  APPEND TO TERM.
8860 NEXT A              \*REPEAT FOR NEXT LITERAL.
8870 RETURN              \*RETURN AND SUBROUTINE.
17000*****************************************************************
17001*        SUBROUTINE TO MINIMIZE FUNCTIONS.
17002*   A FUNCTION IS SAID TO BE MINIMIZED (IN THIS CONTEXT) IF IT
17003*   CONTAINS NO REDUNDANT PRODUCT TERMS.  HENCE THE SUBROUTINE
17004*   REMOVES DUPLICATE TERMS AND TERMS COVERED BY ANOTHER TERM.
17005*        A FUNCTION IS PASSED TO THE SUBROUTINE IN ARRAY H.
17006*   THE REDUCED FUNCTION IS RETURNED IN ARRAY H.
17007*        CALLED BY THE AND SUBROUTINE AND THE OR SUBROUTINE.
17008*****************************************************************
17010 FOR X=1 TO X2-1          \*EXAMINE EACH TERM OF THE FN.
17015         INPUT :H;X,Z$    \*GET A TERM FROM ARRAY H.
17016         IF Z$="." THEN 17220\*ENDOF FUNCTION?
17020         X$=Z$            \*NO. COPY TERM TO X$ FOR PARSING.
17030         *PARSE TERM INTO LITERALS.\GOSUB 7800
17040         FOR Y=1 TO 20    \*COPY LITERALS TO ARRAY D$.
17050              D$(Y)=A$(Y)\*COPY A TERM.
17060              IF A$(Y)="." THEN 17080\*END OF LIST?
17070         NEXT Y           \*NO.  REPEAT FOR NEXT LITERAL.
17080         FOR Y=X+1 TO X2-1\*COMPARE ABOVE TERM TO ALL OTHER TERMS.
17090              INPUT :H;Y,Z$\*GET A TERM FROM ARRAY H.
17091              IF Z$="." THEN 17210\*END OF FN?
17100              X$=Z$       \*COPY TERM TO X$ FOR PARSING.
17110              *PARSE THE TERM INTO LITERALS\GOSUB 7800
17115              F5=1        \*INITIALIZE FLAG INDICATING
17116                          *A REDUNDANT TERM.
17120              FOR B=1 TO 20\*COMPARE LITERAL LISTS OF TWO TERMS.
17122                   IF F5=0 THEN 17200\*CHECK FLAG FOR NO REDUNDANCY.
17125                   IF D$(B)="." THEN 17170\*END OF LIST?
17130                   FOR A=1 TO 20\*SCAN LIST OF 2ND TERM.
17135                        IF A$(A)="." THEN 17165\*END OF LIST?
17140                        IF A$(A)=D$(B) THEN 17160
17141                             *COMPARE LITERALS OF TWO LISTS.
17150                        F5=0 \*LITERALS ARE NOT ALIKE.
17151                             *CLEAR FLAG TO INDICATE NO REDUNDANCY.
17155                   NEXT A    \*REPEAT FOR NEXT LITERAL OF 2ND TERM
17160                   F5=1      \*MATCH FOUND. POSSIBLE REDUNDANCY.
```

```
17165              NEXT B          \*REPEAT FOR NEXT LITERAL OF 1ST TERM
17170              FOR C=Y TO X2-1\*2ND TERM REDUNDANT.  DELETE FROM H.
17180                   INPUT :H;C+1,Z$\*GET A TERM FROM H.
17185                   PRINT :H;C,Z$  \*PLACE TERM IN NEW POSITION.
17190              NEXT C          \*REPEAT FOR NEXT TERM.
17195              *GET NEW 2ND TERM.\GOTO 17090
17200         NEXT Y           \*REPEAT FOR NEXT TERM.
17210 NEXT X                   \*REPEAT FOR NEXT FIRST TERM.
17220 RETURN                   \*RETURN TO CALLING PROGRAM.
20000 END
```

```
10:11 OCT 11,'79 DC/TESTGNDOC.BCARROLL

1*      SSSSS   III   MM MM   L        00000   GGGGG
2*      S        I    MM MM   L        0   0   G
3*      SSSSS    I    M M M   L        0   0   G  GG
4*          S    I    M   M   L        0   0   G   G
5*      SSSSS   III   M   M   LLLLL    00000   GGGGG
6*
7*          PROGRAM:        TESTGNDOC
8*          VERSION:        SIGMA 5
9*          REVISION:       ORIGINAL
10*         DATE:           10/10/79
11*         PROGRAMMER:     B. D. CARROLL
12*
13*****************************************************************
14*
15*          CONTRACT SUPPORT
16*****************************************************************
17*
18*      THIS PROGRAM WAS DEVELOPED FOR NASA MARSHALL FLIGHT
19*      CENTER UNDER CONTRACT NAS8-31572.
20*
21*****************************************************************
22*
23*          MODIFICATION HISTORY
24*
25*****************************************************************
26*
39*
100****************************************************************
101*
102*          PROGRAM DESCRIPTION
103*
104****************************************************************
105*
106*     TESTGN IS A PROGRAM FOR GENERATING TEST SEQUENCES FROM THE
107*     SIMULATION OUTPUT PRODUCED BY THE SIMLOG PROGRAM.
108*     TESTS MAY BE GENERATED FOR CIRCUIT INPUTS FROM A FAULT-FREE
109*     SIMULATION.  TESTS FOR SPECIFIC FAULTS MAY ALSO BE GENERATED
110*     FROM A FAULT SIMULATION OF THE CIRCUIT WITH THE FAULT
111*     INSERTED.
112*
900****************************************************************
901*
902*          DIMENSION STATEMENTS
903*
904****************************************************************
910       DIM N(200,4), I(400,3), G(50,2), P(200,2)
920       DIM A$(51),D$(51)
930       DIM U(1)
940      DIM L$(10),M$(10,2)
950       DIM E$(200)
```

```
1000*****************************************************************************
1001*****************************************************************************
1002*                                                                           *
1003*          START OF MAIN PROGRAM                                            *
1004*                                                                           *
1005*****************************************************************************
1006*****************************************************************************
1007*
1008*   OPEN CIRCUIT DESCRIPTION FILE(NET).
1010 OPEN "NET" TO :3, INPUT UPDATE
1100 H9=400
1110 Y5=100
1200 PRINT"SELECT TEST GENERATION MODE.  <CR>=1."
1205 PRINT"     (1)      UNSPECIFIED FAULTS."
1210 PRINT"     (2)      SPECIFIED FAULTS."
1212 INPUT M5
1215 IF M5<>0 THEN 1220
1217 M5=1
1220 IF M5<=2 THEN 1235
1225 PRINT"INVALID MODE NUMBER ENTERED.  TRY AGAIN."
1230 GOTO 1212
1234    *LOAD CIRCUIT DESCRIPTION FROM FILE NET.
1235 GOSUB 6000
1239    *CONSTRUCT A LIST OF PRIMARY OUTPUT NAMES.
1240 Y=1
1245 FOR X=1 TO N1
1250    IF N(X,4)<>"YES" THEN 1270
1255    M$(Y,1)=N(X,1)
1260    M$(Y,2)=STR(X)
1265    Y=Y+1
1270 NEXT X
1274    *COMPUTE THE NUMBER OF PRIMARY OUTPUT NETS.
1275 O3=Y-1
1279    *CONSTRUCT A LIST OF PRIMARY INPUT NAMES.
1280 Y=1
1285 FOR X=1 TO I0
1290    IF I(X,3)<>"YES" THEN 1305
1292    IF Y=1 THEN 1298
1294    FOR Z=1 TO Y-1
1295        IF L$(Z)=I(X,1) THEN 1305
1296    NEXT Z
1298    L$(Y)=I(X,1)
1300    Y=Y+1
1305 NEXT X
1309    *COMPUTE THE NUMBER OF PRIMARY INPUT NETS.
1310 I3=Y-1
1319    *CHECK FOR MODE SELECTION(M5=2: SPECIFIED FAULTS).
1320 IF M5=2 THEN 1500
1349    *BEGIN PRIMARY CONTROL LOOP FOR UNSPECIFIED FAULT MODE.
1350 FOR X6=1 TO O3
1354    *SELECT A PRIMARY OUTPUT NET.
1355    O$=M$(X6,1)
```

```
1356    X5=VAL(M$(X6,2))
1360    FOR Y6=1 TO I3
1364        *SELECT A PRIMARY INPUT NET.
1365        W$=L$(Y6)
1370        FOR T=1 TO T0
1372            *SELECT A TIME TAG.
1373            T$=STR(T)
1374            T$=T$(:2)
1375            *COMPUTE CORRESPONDING TEST FN.\GOSUB 2020
1377            *PRINT TEST FUNCTION.    !GOSUB 11009
1380        NEXT T
1385    NEXT Y6
1390 NEXT X6
1395 GOTO 1620
1500 PRINT"ENTER THE FAULTED NET NAME";
1510 INPUT W$
1520 PRINT"ENTER THE FAULT CONDITION(S0/S1)";
1530 INPUT S$
1535 IF S$="S0" THEN 1538
1536 T$="!"
1537 GOTO 1540
1538 T$="*"
1539 *BEGIN PRIMARY CONTROL LOOP FOR SPECIFIED FAULTS MODE.
1540 FOR X6=1 TO 03
1549    *SELECT A PRIMARY OUTPUT NET.
1550    O$=M$(X6,1)
1560    X5=VAL(M$(X6,2))
1569    *COMPUTE THE TEST FUNCTION.
1570    GOSUB 2020
1579    *PRINT TEST FUNCTION HEADER INFO.
1580    PRINT"OUTPUT-LINE=";O$
1590    PRINT"FAULT=";W$+"/"+S$
1599    *PRINT THE TEST FUNCTION.
1600    GOSUB 11030
1610 NEXT X6
1620 CLOSE :1\CLOSE :2\CLOSE :3\CLOSE :4
1630 CHAIN LINK "SIM1EXEC"
2000*************************************************************************
2001*************************************************************************
2002*          SUBROUTINES                                                  *
2003*************************************************************************
2004*
2005*-----------------------------------------------------------------------
2006*************************************************************************
2007*      SUBROUTINE FOR COMPUTING A TEST FUNCTION.
2008*************************************************************************
2009*-----------------------------------------------------------------------
2020 *RETRIEVE OUTPUT FUNCTION PAIR.\GOSUB 3000
2049 *COMPUTE THE FIRST REDUCED FUNCTION.
2050 FOR X=1 TO 200
2055    INPUT :F;X,X$
2060    IF X$="." THEN 2150
```

```
2065     Y=1
2070     GOSUB 4000
2072     IF Z9=0 THEN 2095
2080     IF Z9=1 THEN 2100
2090     IF X$(:Z9-1,1)>"Z" THEN 2100
2091     IF Z9+L2>L1-L2+1 THEN 2095
2092     Y=Z9+L2
2093     GOTO 2070
2095     X$=""
2096     PRINT :F;X,X$
2098     GOTO 2110
2100     IF X$<>W$+1$ THEN 2105
2102     PRINT :F;1,"1"
2103     PRINT :F;2,"."
2104     GOTO 2300
2105     IF Z9=1 THEN 2108
2106     PRINT :F;X,X$(:1,Z9-1)+X$(:Z9+L2)
2107     GOTO 2110
2108     PRINT :F;X,X$(:Z9+L2)
2110 NEXT X
2150 PRINT :H;1,""
2153 Y=1
2155 FOR X=1 TO 200
2165     INPUT :F;X,X$
2166     IF X$="" THEN 2190
2170     PRINT :H;Y,X$
2172     IF X$="." THEN 2200
2175     Y=Y+1
2190 NEXT X
2200 INPUT :H;1,X$
2201 IF X$="." THEN 2250
2205 FOR X=1 TO 200
2210     INPUT :H;X,X$
2211     PRINT :F;X,X$
2215     IF X$="." THEN 2300
2220 NEXT X
2250 PRINT :F;1,"0"
2260 PRINT :F;2,"."
2299 *COMPUTE THE SECOND REDUCED FUNCTION
2300 FOR X=1 TO 200
2305     INPUT :G;X,X$
2310     IF X$="." THEN 2332
2311     Y=1
2312     GOSUB 4500
2313     IF Z9=0 THEN 2320
2314     IF Z9=1 THEN 2322
2315     IF X$(:Z9-1,1)>"Z" THEN 2322
2316     IF Z9+L2>L1-L2+1 THEN 2320
2317     Y=Z9+L2
2318     GOTO 2312
2320     PRINT :G;X,""
2321     GOTO 2330
```

```
2322     IF X$<>N$+"-"+T$ THEN 2326
2323     PRINT :G;1,"1"
2324     PRINT :G;2,"."
2325     GOTO 2400
2326     IF Z9=1 THEN 2329
2327     PRINT :G;X,X$(:1,Z9-1)+X$(:Z9+L2)
2328     GOTO 2330
2329     PRINT :G;X,X$(:Z9+L2)
2330 NEXT X
2332 PRINT :H;1,""
2335 Y=1
2340 FOR X=1 TO 200
2345     INPUT :G;X,X$
2346     IF X$="" THEN 2365
2350     PRINT :H;Y,X$
2355     IF X$="." THEN 2370
2360     Y=Y+1
2365 NEXT X
2370 IF Y=1 THEN 2394
2375 FOR X=1 TO 200
2380     INPUT :H;X,X$
2381     PRINT :G;X,X$
2385     IF X$="." THEN 2400
2390 NEXT X
2394 PRINT :G;1,"0"
2396 PRINT :G;2,"."
2399 *COMPUTE THE FIRST AND RESULTANT.
2400 GOSUB 8000
2410 FOR X=1 TO 200
2415     INPUT :H;X,X$
2416     E$(X)=X$
2420     IF X$="." THEN 2430
2425 NEXT X
2430 *RETRIEVE THE OUTPUT FUNCTION PAIR ONCE AGAIN.
2431 *COMPUTE THE THIRD REDUCED FUNCTION.
2440 GOSUB 3000
2442 FOR X=1 TO 200
2444 INPUT :F;X,X$
2446 IF X$="." THEN 2480
2448 Y=1
2450     GOSUB 4500
2451     IF Z9=0 THEN 2462
2452     IF Z9=1 THEN 2468
2454     IF X$(:Z9-1,1)>"Z" THEN 2468
2456     IF Z9+L2>L1-L2+1 THEN 2462
2458     Y=Z9+L2
2460 GOTO 2450
2462 X$=""
2464 PRINT :F;X,X$
2466     GOTO 2479
2468     IF X$<>N$+"-"+T$ THEN 2475
2470 PRINT :F;1,"1"
```

```
2472 PRINT :F;2,"."
2474 GOTO 2516
2475     IF Z9=1 THEN 2479
2476     PRINT :F;X,X$(:1,Z9-1)+X$(:Z9+L2)
2477     GOTO 2479
2478     PRINT :F;X,X$(:Z9+L2)
2479 NEXT X
2480 PRINT :H;1,""
2482 Y=1
2484 FOR X=1 TO 200
2486 INPUT :F;X,X$
2488 IF X$="" THEN 2496
2490 PRINT :H;Y,X$
2492 IF X$="." THEN 2498
2494 Y=Y+1
2496 NEXT X
2498 INPUT :H;1,X$
2500 IF X$="." THEN 2512
2502 FOR X=1 TO 200
2504 INPUT :H;X,X$
2506 PRINT :F;X,X$
2508 IF X$="." THEN 2516
2510 NEXT X
2512 PRINT :F;1,"0"
2514 PRINT :F;2,"."
2515 *COMPUTE THE FOURTH REDUCED FUNCTION.
2516 FOR X=1 TO 200
2518 INPUT :G;X,X$
2520     IF X$="." THEN 2552
2522 Y=1
2524     GOSUB 4000
2525     IF Z9=0 THEN 2536
2526     IF Z9=1 THEN 2540
2528     IF X$(:Z9-1,1)>"Z" THEN 2540
2530     IF Z9+L2>L1-L2+1 THEN 2536
2532     Y=Z9+L2
2534 GOTO 2524
2536 PRINT :G;X,""
2538 GOTO 2550
2540     IF X$<>W$+I$ THEN 2546
2542 PRINT :G;1,"1"
2544 PRINT :G;2,"."
2545     GOTO 2586
2546     IF Z9=1 THEN 2549
2547     PRINT :G;X,X$(:1,Z9-1)+X$(:Z9+L2)
2548     GOTO 2550
2549     PRINT :G;X,X$(:Z9+L2)
2550 NEXT X
2552 PRINT :H;1,""
2554 Y=1
2556 FOR X=1 TO 200
2558 INPUT :G;X,X$
```

```
2560 IF X$="" THEN 2568
2562    PRINT :H;Y,X$
2564 IF X$="." THEN 2570
2566 Y=Y+1
2568 NEXT X
2570 IF Y=1 THEN 2582
2572 FOR X=1 TO 200
2574 INPUT :H;X,X$
2576 PRINT :G;X,X$
2578 IF X$="." THEN 2586
2580 NEXT X
2582 PRINT :G;1,"0"
2584 PRINT :G;2,"."
2585 *COMPUTE THE SECOND AND RESULTANT.
2586 GOSUB 8000
2750 FOR X=1 TO 200
2755    PRINT :F;X,E$(X)
2760    IF E$(X)="." THEN 2770
2765 NEXT X
2770 FOR X=1 TO 200
2775    INPUT :H;X,X$
2776    PRINT :G;X,X$
2780    IF X$="." THEN 2790
2785 NEXT X
2789 *COMPUTE THE OR RESULTANT.
2790 GOSUB 7000
2792 IF MS=2 THEN 2999
2795 FOR X=1 TO 200
2800    INPUT :H;X,X$
2801    PRINT :F;X,X$
2805    IF X$="." THEN 2815
2810 NEXT X
2815 PRINT :G;1,W$+T$
2820 PRINT :G;2,W$+"-"+T$
2825 PRINT :G;3,"."
2829 *COMPUTE THE TEST FUNCTION.
2830 GOSUB 8000
2999 RETURN
3000*****************************************************************
3001*     ROUTINE TO RETRIEVE OUTPUT FUNCTIONS.
3002*****************************************************************
3009 *RETRIEVE THE ONE-FUNCTION.
3010 J=4
3020 K=1
3030 GOSUB 5000
3039 *RETRIEVE THE ZERO-FUNCTION.
3040 J=3
3050 K=0
3060 GOSUB 5000
3070 RETURN
4000*****************************************************************
4001*        SUBROUTINE FOR SEARCHING W$+T$ FOR THE SUBSTRING X$.
```

```
4002*    POSITION OF X$ IS RETURNED AS Z9.  Z9=0 IS RETURNED IF
4003*    X$ IS NOT A SUBSTRING OF W$+T$.
4004 *INITIALIZE THE POINTER.
4005 Z9=0
4010 L1=LEN(X$)
4015 L2=LEN(W$+T$)
4019 *SCAN X$ STARTING AT Y FOR W$+T$.
4020 FOR Z=Y TO L1
4025     IF X$(:Z,L2)=W$+T$ THEN 4040
4030 NEXT Z
4035 RETURN
4039 *LOAD POINTER WITH POSITION OF SUBSTRING.
4040 Z9=Z
4045 RETURN
4500*****************************************************************
4501*         SUBROUTINE FOR SEARCHING STRING W$+"-"+T$ FOR SUBSTRING
4502*    X$.  POSITION OF X$ IS RETURNED AS Z9.  Z9=0 IS RETURNED IF X$
4503*    IS NOT A SUBSTRING OF W$+"-"+T$.
4504 *INITIALIZE THE POINTER.
4505 Z9=0
4510 L1=LEN(X$)
4515 L2=LEN(W$+"-"+T$)
4519 *SCAN X$ STARTING AT Y FOR W$+"-"+T$.
4520 FOR Z=Y TO L1
4525     IF X$(:Z,L2)=W$+"-"+T$ THEN 4540
4530 NEXT Z
4535 RETURN
4539 *LOAD POINTER WITH POSITION OF SUBSTRING.
4540 Z9=Z
4545 RETURN
5000*****************************************************************
5001*         SUBROUTINE TO TRANSFER FUNCTION FROM
5002*         FUNCTION FILE TO FUNCTION ARRAY.
5003*****************************************************************
5005 INPUT=0
5006 *DEFINE FILE CHANNEL FUNCTIONS.
5007 E=J-2*S*J+5*S
5010 F=1-3*S+S*J
5015 G=7+S
5020 H=7-3*S-J+J*S
5024 *CLOSE ALL FILES.
5025 CLOSE :1\CLOSE :2\CLOSE :3\CLOSE :4
5029 *OPEN FILE CONTAINING FUNCTION TO BE RETRIEVED.
5030 IF S=1 THEN 5050
5035 IF J=4 THEN 5045
5040 OPEN "NEW0" TO :3, INPUT UPDATE
5041 OPEN "FARRAY" TO :1,INPUT UPDATE
5042 OPEN "GARRAY" TO :2,INPUT UPDATE
5043 OPEN "HARRAY" TO :4,INPUT UPDATE
5044 GOTO 5140
5045 OPEN "NEW1" TO :4, INPUT UPDATE
5046 OPEN "HARRAY" TO :3,INPUT UPDATE
```

```
5047 OPEN "FARRAY" TO :1,INPUT UPDATE
5048 OPEN "GARRAY" TO :2,INPUT UPDATE
5049 GOTO 5140
5050 IF J=4 THEN 5060
5052 OPEN "FAPRAY" TO :1, INPUT UPDATE
5053 OPEN "GARRAY" TO :3,INPUT UPDATE
5054 OPEN "HARRAY" TO :4,INPUT UPDATE
5055 OPEN "OLDO" TO :2, INPUT UPDATE
5056 GOTO 5140
5060 OPEN "OLD1" TO :1, INPUT UPDATE
5061 OPEN "FARRAY" TO :2,INPUT UPDATE
5062 OPEN "GARRAY" TO :3,INPUT UPDATE
5063 OPEN "HARRAY" TO :4,INPUT UPDATE
5140 V=X5
5149 *TRANSFER THE FUNCTION TO ARRAY G.
5150 INPUT :E;Y$*(V-1)+1,X$
5160 FOR X=1 TO 1000
5170      INPUT :E,Y$
5180 PRINT :G;X,Y$
5190   IF Y$="." THEN 5260
5200 NEXT X
5260 IF K<>1 THEN 5300
5269 *TRANSFER THE FUNCTION TO FARRAY.
5270 FOR Y=1 TO X
5280 INPUT :G;Y,Y$
5285 PRINT :F;Y,Y$
5290 NEXT Y
5300 INPUT=$
5310 RETURN
6000*************************************************************************
6001*     SUBROUTINE TO INPUT CIRCUIT DESCRIPTION FROM FILE.                *
6002*************************************************************************
6010 INPUT=$
6015 I0=1\                  ***INITIALIZE I ARRAY POINTER.
6017 N1=0\                  ***INITIALIZE N ARRAY POINTER.
6020 Z$=""\                 ***CLEAR Z$.
6022 INPUT :3,Z$\           ***INPUT ELEMENT DESCRIPTION FROM FILE.
6023 PRINT Z$\              ***PRINT ELEMENT DESCRIPTION.
6025 IF Z$="." THEN 6430\*LOOK FOR END OF ELEMENT ENTRIES.
6030 N1=N1+1\               ***INC N ARRAY POINTER.
6034***** BEGIN PARSING OF Z$ **********
6035 B$=""\                 ***CLEAR B$ FOR NEXT FIELD.
6040 X=1\                   ***INITIALIZE POINTER TO Z$.
6050 IF Z$(:X,1)=" " THEN 6090\*END OF ELEMENT-NAME FIELD?
6060 B$=B$+Z$(:X,1)\        ***END NOT REACHED.  ADD TO NAME.
6070 X=X+1\                 ***INC POINTER TO Z$.
6080 ***REPEAT FOR NEXT CHARACTER IN Z$.\GOTO 6050
6090 N(N1,1)=B$\            ***WHEN NAME COMPLETE, PLACE IN N ARRAY.
6100 N(N1,4)="NO"\          ***LABEL ALL ELEMENTS AS NO OUTPUT.
6110 B$=""\                 ***CLEAR B$ FOR NEXT FIELD.
6120 X=X+1\                 ***INC POINTER TO Z$.
6130 IF Z$(:X,1)=" " THEN 6170\*END OF ELEMENT TYPE FIELD?
```

```
6140 B$=B$+Z$(:X,1)\    ***END NOT REACHED, BUILD NAME.
6150 X=X+1\             ***INC POINTER TO Z$.
6160***REPEAT FOR NEXT CHARACTER IN Z$.\GOTO 6130
6170 N(N1,2)=B$\        ***WHEN NAME COMPLETE, PLACE IN N ARRAY.
6175 Z=1\               ***INITIALIZE COUNT OF ELEMENT INPUTS.
6177 P(N1,2)=I0\        ***LOAD POINTER TO INPUT LIST.
6180 X=X+1\             ***INC POINTER TO Z$.
6190 B$=""\             ***CLEAR B$ FOR NEXT FIELD.
6200 IF Z$(:X,1)=" " THEN 6250\*END OF INPUT NAME SUBFIELD?
6210 B$=B$+Z$(:X,1)\    ***END NOT REACHED.  RECONSTRUCT NAME.
6230 X=X+1\             ***INC POINTER TO Z$.
6240 ***REPEAT FOR NEXT CHARACTER IN Z$.\GOTO 6200
6250 I(I0,1)=B$\        ***WHEN NAME COMPLETE, PLACE IN I ARRAY.
6260 Z=Z+1\             ***INC COUNT OF ELEMENT INPUTS.
6270 I0=I0+1\           ***INC POINTER TO I ARRAY.
6280 L=LEN(Z$)\         ***COMPUTE LENGTH OF Z$.
6285 IF X<L THEN 6180\****IF END NOT REACHED, GET NEXT INPUT.
6290 P(N1,1)=Z-1\       ***WHEN END REACHED, PLACE # OF INPUTS IN P.
6300 ***REPEAT FOR NEXT ELEMENT DESCRIPTION.\GOTO 6020
6430 FOR  X=1 TO I0-1\ ***ASSIGN NUMERIC CODE TO ALL INPUTS
6431     FOR Y=1 TO N1\ ***EXCEPT PRIMARY INPUTS.
6432          IF I(X,1)<>N(Y,1) THEN 6436
6433          Z$=STR(Y)
6434          I(X,3)=Z$
6435          GOTO 6437
6436     NEXT Y
6437 NEXT X
6442 I0=I0-1\           ***COMPUTE THE NUMBER OF ROWS IN I.
6445 INPUT :3,Z$\       ***GET PRIMARY INPUT NAME FROM FILE.
6446 PRINT Z$\          ***PRINT PRIMARY INPUT NAME.
6448 IF Z$=" " THEN 6465\*LOOK FOR END OF PRIMARY INPUT ENTRIES.
6450 FOR X=1 TO I0\     ***SCAN I ARRAY FOR PRIMARY INPUT NAMES.
6455     IF Z$<>I(X,1) THEN 6460
6457     I(X,3)="YES"\  ***FLAG WITH YES WHEN FOUND.
6460 NEXT X
6462****REPEAT FOR NEXT INPUT NAME.\GOTO 6445.
6465 INPUT :3,Z$\       ***GET PRIMARY OUTPUT NAME FROM FILE.
6466 PRINT Z$\          ***PRINT PRIMARY OUTPUT NAME.
6470 IF Z$=" " THEN 6500\*LOOK FOR END OF OUTPUT NAME ENTRIES.
6472 FOR X=1 TO N1\     ***SCAN N ARRAY FOR PRIMARY OUTPUT NAMES.
6475     IF Z$<>N(X,1) THEN 6480
6477     N(X,4)="YES"\  ***FLAG WITH YES WHEN FOUND.
6480 NEXT X
6485 ***REPEAT FOR NEXT OUTPUT NAME.\GOTO 6465
6500 Q1=0
6510 INPUT :3,Z$\       ***GET XC ELEMENT PAIR FROM FILE.
6511 PRINT Z$\          ***PRINT ELEMENT PAIR.
6520 IF Z$=" " THEN 6700\*LOOK FOR END OF XC PAIR ENTRIES.
6530 Q1=Q1+1\           ***INC POINTER TO Q ARRAY.
6540 L=LEN(Z$)\         ***COMPUTE LENGTH OF Z$.
6550 FOR X=1 TO L\      ***BEGIN PARSING Z$.
6560     IF Z$(:X,1)=" " THEN 6580\*END OF FIRST NAME IN PAIR?
```

```
6570 NEXT X
6580 B$=Z$(:1,X-1)\      ***LET B$ = FIRST NAME.
6590 C$=Z$(:X+1)\        ***LET C$ = SECOND NAME.
6640 FOR X1=1 TO N1\     ***SCAN N ARRAY FOR NAMES OF XC ELEMENTS.
6641    IF B$=N(X1,1) THEN 6645
6643 NEXT X1
6645 Z$=STR(X1)\         ***GENERATE NUMERIC CODE FOR ELEMENT.
6646 Q(Q1,1)=Z$\         ***PLACE CODE IN Q ARRAY.
6680 FOR X1=1 TO N1\     ***SCAN N ARRAY FOR SECOND NAME IN XC PAIR.
6681    IF C$=N(X1,1) THEN 6685
6682 NEXT X1
6685 Z$=STR(X1)\         ***GENERATE NUMERIC CODE.
6686 Q(Q1,2)=Z$\         ***PLACE CODE IN Q ARRAY.
6690 ***REPEAT FOR NEXT PAIR OF XC ELEMENTS.\GOTO 6510
6700 RETURN\             ***RETURN TO MAIN PROGRAM (1340).
7000*************************************************************************
7001*        SUBROUTINE FOR ORING TWO FUNCTIONS.
7002*   ONE FUNCTION MUST BE IN ARRAY F AND THE SECOND IN ARRAY G.
7003*   THE RESULTANT FUNCTION IS PLACED IN ARRAY H.
7004*   CALLED BY THE MAIN PROGRAM ONLY.
7005*   CALLS THE SORT SUBROUTINE AND THE MINIMIZATION SUBROUTINE.
7006*************************************************************************
7010 INPUT :F;1,X$       \*GET FIRST TERM OF FUNCTION FROM F.
7011 IF X$<>"0" THEN 7035\*CHECK FOR ZERO FUNCTIONAL VALUE.
7015 INPUT :G;1,Y$       \*GET FIRST TERM OF FUNCTION FROM G.
7016 IF Y$<>"0" THEN 7075\*CHECK FOR ZERO FUNCTIONAL VALUE.
7020 PRINT :H;1,"0"      \*BOTH INPUTS ARE ZERO.  RESULT IS ZERO.
7025 PRINT :H;2,"."      \*LOAD END-OF-FUNCTION MARK.
7030 RETURN              \*RETURN TO MAIN PROGRAM.
7035 INPUT :G;1,Y$       \*GET FIRST TERM OF FUNCTION FROM G.
7036 IF Y$<>"0" THEN 7110\*FIRST INPUT NOT ZERO, IS THE SECOND?
7040 FOR X=1 TO H9       \*SECOND INPUT IS ZERO.
7045    INPUT :F;X;X$\*RESULTANT IS EQUAL TO FIRST FUNCTION.
7046    PRINT :H;X,X$
7050    IF X$="." THEN 7060\*LOOK FOR END OF FUNCTION.
7055 NEXT X              \*REPEAT FOR NEXT TERM.
7060 *SORT THE TERMS.    \GOSUB 7200
7065 *REDUCE THE RESULTANT FN.\GOSUB 17000
7070 RETURN              \*RETURN TO MAIN PROGRAM.
7075 FOR X=1 TO H9       \*FIRST FN=0.  SECOND FN<>0.
7080    INPUT :G;X;Y$    \*RESULTANT FN = SECOND FN.
7081    PRINT :H;X,Y$
7085    IF Y$="." THEN 7095\*LOOK FOR END OF FUNCTION.
7090 NEXT X              \*REPEAT FOR NEXT TERM.
7095 *SORT THE TERMS.    \GOSUB 7200
7100 *REDUCE THE RESULTANT FN.\GOSUB 17000
7105 RETURN              \*RETURN TO MAIN PROGRAM.
7110 INPUT :F;1,X$       \*GET FIRST TERM OF FIRST FN.
7111 IF X$<>"1" THEN 7130\*CHECK FOR ONE FUNCTIONAL VALUE.
7115 PRINT :H;1,"1"      \*FIRST FUNCTION IS ONE. RESULT IS ONE.
7120 PRINT :H;2,"."      \*LOAD END-OF-FILE MARK.
7125 RETURN              \*RETURN TO MAIN PROGRAM.
```

```
7130 INPUT :G;1,Y$      \*GET FIRST TERM OF SECOND FUNCTION.
7131 IF Y$<>"1" THEN 7150\*CHECK FOR ONE FUNCTIONAL VALUE.
7135 PRINT :H;1,"1"     \*SECOND FN = 1.  RESULTANT FN =1.
7140 PRINT :H;2,"."     \*LOAD END-OF-FUNCTION MARK.
7145 RETURN             \*RETURN TO MAIN PROGRAM.
7149 *NEITHER INPUT FN IS ZERO OR ONE.  COMPUTE THE RESULT.
7150 Y=1                \*INITIALIZE POINTER TO H ARRAY.
7155 FOR X=1 TO 99      \*COPY FIRST FN INTO H ARRAY.
7160    INPUT :F;X,X$   \*GET TERM OF FN.
7161    PRINT :H;Y,X$   \*PLACE TERM IN H.
7165    INPUT :F;X+1,X$\*GET NEXT TERM OF FN.
7166    IF X$="." THEN 7180\*END OF FN?
7170    Y=Y+1           \*NO.  INC POINTER TO H.
7175 NEXT X             \*INC POINTER TO F.
7180 Y=Y+1              \*YES.  INC POINTER TO H.
7185 FOR X=1 TO 99      \*COPY G INTO H FOLLOWING F.
7187    INPUT :G;X,Y$   \*GET TERM OF G.
7188    PRINT :H;Y,Y$   \*PLACE TERM IN H.
7189    IF Y$="." THEN 7195\*LOOK FOR END OF FN G.
7191    Y=Y+1           \*END NOT REACHED.  INC POINTER TO H.
7193 NEXT X             \*INC POINTER TO G.
7195 *SORT TERMS.       \GOSUB 7200
7197 *REDUCE RESULTANT FUNCTION.\GOSUB 17000
7199 RETURN             \*RETURN TO MAIN PROGRAM.
7200*************************************************************************
7201*        SUBROUTINE TO SORT THE TERMS OF A FUNCTION.
7202*        FUNCTION MUST BE STORED IN ARRAY H BEFORE
7203*        SORTING RULES: LENGTH OF TERM, ALPHABETICAL, NUMERICAL.
7204*        CALLED BY OR AND AND SUBROUTINES.
7205*        CALLS NO SUBROUTINES.
7206*************************************************************************
7220 FOR X=1 TO 100          \*COUNT NUMBER OF TERMS IN FN.
7230    INPUT :H;X,Z$        \*GET A TERM.
7231    IF Z$="." THEN 7250 \*END OF FUNCTION?
7240 NEXT X                  \*NO.
7249 *X = THE NUMBER OF TERMS IN H PLUS 1.
7250 IF X=2 THEN 7500        \*CHECK FOR SINGLE TERM FN.
7252 X0=X                    \*SAVE X FOR USE AS SORT LOOP LIMITER.
7255 X1=X                    \*SAVE X FOR STORE LOOP LIMITER.
7260 FOR X=1 TO X0-2    \*USE A BUBBLE SORT TO ORDER THE TERMS.
7270    INPUT :H;X,X$   \*GET FIRST TERM OF H.
7272    INPUT :H;X+1,Y$ \*GET ADJACENT TERM OF H.
7274    IF X$<Y$ THEN 7310\*COMPARE THE TERMS.
7280    *REVERSE THE ORDER OF THE TWO TERMS.
7290    PRINT :H;X,Y$
7300    PRINT :H;X+1,X$
7310 NEXT X             \*REPEAT FOR NEXT PAIR OF TERMS.
7319 * ONE TERM HAS BEEN PLACED IN ITS PROPER PLACE.
7320 X0=X0-1            \*DECREMENT SORT LOOP LIMITER.
7330 IF X0>2 THEN 7260 \*HAVE ALL TERMS BEEN ORDERED?
7340 X0=X1              \*YES.  RESTORE X0.
7344 *DUPLICATE TERMS WILL NOW BE REMOVED.
```

```
7345 FOR X=1 TO X1      \*COMPARE ADJACENT TERMS FOR EQUALITY.
7347     INPUT :H;X,X$  \*GET A TERM.
7348     IF X$="." THEN 7420\*END OF FUNCTION?
7350     INPUT :H;X+1,Y$\*NO.  GET ADJACENT TERM.
7351     IF X$=Y$ THEN 7360\*ARE TERMS IDENTICAL?
7355     *NO.  GET NEXT PAIR.\GOTO 7410
7360     X0=X0-1        \*YES.  DECREMENT TERM COUNT.
7365     FOR Y=X+1 TO X1-1\*REMOVE DUPLICATE TERM AND COMPACT H.
7370          INPUT :H;Y+1,X$\*GET TERM FROM H.
7371          PRINT :H;Y,X$\*MOVE TERM TO NEW POSITION IN H.
7380          IF X$="." THEN 7410\*END OF FUNCTION?
7390     NEXT Y         \*REPEAT FOR NEXT TERM.
7410 NEXT X             \*REPEAT FOR NEXT PAIR OF TERMS.
7420 FOR X=1 TO X0-2    \*SORT TERMS BY LENGTH USING BUBBLE SORT.
7430     INPUT :H;X,X$
7431     INPUT :H;X+1,Y$
7432     L1=LEN(X$)
7433     L2=LEN(Y$)
7434     IF L1<=L2 THEN 7470\*IF TRUE, DO NOT REORDER.
7440     PRINT :H;X,Y$  \*REORDER.
7450     PRINT :H;X+1,X$
7470 NEXT X             \*REPEAT FOR NEXT PAIR.
7480 X0=X0-1            \*DECREMENT LOOP LIMIT FOR NEXT PASS.
7490 IF X0>2 THEN 7420  \*SORTING COMPLETE?
7500 RETURN             \*YES.  RETURN TO CALLING PROGRAM.
7800*****************************************************************
7801*     SUBROUTINE TO DISASSEMBLE A PRODUCT TERM INTO ITS
7802*   LITERAL COMPONENTS.  A LITERAL IS A VARIABLE NAME OR A
7803*   COMPLEMENTED VARIABLE NAME TOGETHER WITH ITS TIME TAG
7804*   OR FAULT SYMBOL.
7805*   TERM IS PASSED TO SUBROUTINE IN X$.  LITERALS RETURNED
7806*   IN A$ ARRAY.  CALLED BY LITERAL SORT SUBROUTINE.
7807*****************************************************************
7810 F5=0               \*CLEAR SPECIAL CHARACTER FLAG.
7815 Z$=""              \*CLEAR DUMMY STRING VARIABLE.
7820 A=1                \*INITIALIZE POINTER TO A$ ARRAY.
7830 L=LEN(X$)          \*COMPUTE LENGTH OF TERM.
7835 FOR X3=1 TO L      \*BEGIN DISASSEMBLY OF TERM.
7840     IF X$(:X3,1)="-" THEN 7900\*LOOK FOR SPECIAL CHARS (-).
7842     IF X$(:X3,1)>"Z" THEN 7900\*LOOK FOR TIME TAG (NUMERAL).
7844     IF X$(:X3,1)<"A" THEN 7900\*LOOK FOR FAULT SYMBOLS (!/*).
7850     IF F5<>1 THEN 7910\*END OF LITERAL REACHED?
7860     A$(A)=Z$       \*YES.  PLACE LITERAL IN A$ ARRAY.
7861                     *END OF LITERAL IS INDICATED WHEN A SPECIAL
7862                     *CHARACTER IS FOLLOWED BY A LETTER.
7863                     *THE LETTER IS THE FIRST CHARACTER OF THE
7864                     *NEXT LITERAL.
7870     Z$=X$(:X3,1)   \*BEGIN CONSTRUCTING NEXT LITERAL.
7880     F5=0           \*CLEAR SPECIAL CHARACTER FLAG.
7890     A=A+1          \*INC POINTER TO A$ ARRAY.
7895     GOTO 7920
7896                     *CONTINUE PROCESSING TERM.
```

```
7900     F%=1              \*SET SPECIAL CHARACTER FLAG.
7910     Z$=Z$+X$(:X3,1)\*CONSTRUCT LITERAL.
7920 NEXT X3               \*REPEAT FOR NEXT CHARACTER OF TERM.
7930 A$(A)=Z$              \*LOAD LAST LITERAL OF TERM IN A$ ARRAY.
7940 A$(A+1)="."           \*LOAD END-OF-LITERAL MARK.
7950 RETURN                \*RETURN TO LITERAL SORT SUBROUTINE.
8000*************************************************************************
8001*          SUBROUTINE TO PERFORM THE LOGICAL AND OF
8002*          TWO FUNCTIONS.
8003*          THE INPUT FUNCTIONS ARE PASSED IN ARRAYS F AND G.
8004*     THE RESULTANT FUNCTION IS RETURNED IN ARRAY H.
8005*          CALLED BY MAIN PROGRAM (SIM2) AND RACE PROGRAM.
8006*          CALLS ZERO PRODUCT DETECT SUBR (8250), LITERAL SORT
8007*     SUBR(8650), TERM SORT SUBR(7200), AND MINIMIZATION SUBR(17000)
8008*************************************************************************
8010 INPUT :F;1,X$              \*CHECK FOR ZERO FN VALUE.
8011 IF X$="0" THEN 8030
8020 INPUT :G;1,Y$              \*CHECK FOR ZERO FN VALUE.
8021 IF Y$<>"0" THEN 8050
8030 PRINT :H;1,"0"             \*RESULT IS ZERO.
8040 PRINT :H;2,"."
8045 RETURN                     \*RETURN TO CALLING PROGRAM.
8050 IF X$="1" THEN 8058        \*CHECK FOR ONE VALUED FUNCTION.
8051 IF Y$<>"1" THEN 8068       \*CHECK FOR ONE VALUED FUNCTION.
8052 FOR X=1 TO H9              \*2ND FN ONLY IS ONE VALUED.  RESULT = F.
8053     INPUT :F;X,X$          \*GET TERM OF F.
8054     PRINT :H;X,X$          \*PLACE IN H ARRAY.
8055     IF X$="." THEN 8160    \*END OF FUNCTION?
8056 NEXT X                    \*REPEAT FOR NEXT TERM.
8057 *PROCEED TO FUNCTION REDUCTION STEP.\GOTO 8160
8058 IF Y$<>"1" THEN 8062\*1-ST FN IS 1-VALUED. IS THE 2ND FN 1-VALUED?
8059 PRINT :H;1,"1"             \*BOTH FNS ARE 1-VALUED.  RESULT IS 1-VALUED.
8060 PRINT :H;2,"."
8061 RETURN                     \*RETURN TO CALLING PROGRAM.
8062 FOR X=1 TO H9              \*1ST FN IS 1-VALUED.  2ND IS NOT.
8063     INPUT :G;X,Y$          \*RESULT = G.
8064     PRINT :H;X,Y$
8065     IF Y$="." THEN 8160\*END OF FUNCTION?
8066 NEXT X
8067 *PROCEED TO MINIMIZATION STEP.\GOTO 8160
8068 W=1          \*NEITHER INPUT IS CONSTANT-VALUED. INIT POINTER TO H.
8069 PRINT :H;1,"0"             \*INITIALIZE H TO 0-VALUE.
8070 PRINT :H;2,"."
8075 FOR X=1 TO H9              \*BEGIN PRODUCT-TERM FORMATION LOOP.
8080     INPUT :F;X,X$          \*GET TERM FROM F.
8090     FOR Y=1 TO H9          \*BEGIN LOOP FOR ACCESS TO G.
8091          INPUT :G;Y,Y$\*GET TERM FROM G.
8092          Z$=X$+Y$          \*FORM PRODUCT OF TERMS.
8093          PRINT :H;W,Z$\*PLACE RESULT IN H.
8100          *CHECK FOR ZERO PRODUCT.\GOSUB 8250
8110          INPUT :G;Y+1,Y$\*CHECK G FOR END OF FUNCTION.
8111          IF Y$="." THEN 8130
```

```
8120     NEXT Y              \*NOT END OF G.  GET NEXT TERM.
8130     INPUT :F;X+1,X$     \*CHECK F FOR END OF FUNCTION.
8131     IF X$="." THEN 8150
8140 NEXT X                  \*NOT END OF F.  GET NEXT TERM.
8150 IF W<>1 THEN 8155       \*END OF BOTH FNS REACHED.  0-VALUE RESULT?
8152 PRINT :H;1,"0"          \*YES.  LOAD 0-VALUE IN H.
8153 RETURN                  \*RETURN TO CALLING PROGRAM.
8155 PRINT :H;W,"."          \*NO.  LOAD END-OF-FUNCTION MARK.
8160 FOR X=1 TO H9           \*BEGIN MIN STEP BY ORDERING LITS IN EACH TERM.
8165     INPUT :H;X,X$       \*GET A TERM FROM H.
8167     IF X$="." THEN 8190\*END OF FUNCTION?
8170     *NO.  ORDER LITERALS.\GOSUB 8650
8175     PRINT :H;X,Z$       \*REPLACE ORDERED TERM.
8180 NEXT X                  \*REPEAT FOR NEXT TERM.
8190 *ORDER TERMS OF THE FUNCTION.\GOSUB 7200
8200 *REMOVE REDUNDANT TERMS.\GOSUB 17000
8210 RETURN                  \*RETURN TO CALLING PROGRAM.
8250*************************************************************************
8251*          SUBROUTINE TO CHECK FOR ZERO PRODUCT TERM.
8252*          TERM TO BE CHECKED WAS FORMED BY THE AND SUBROUTINE
8253*     AND WAS PLACED IN THE H ARRAY.  THE TERM IS RETRIEVED FROM H
8254*     FOR PROCESSING.  IF THE TERM IS ZERO, THE POINTER (W) TO THE
8255*     H ARRAY IS NOT INCREMENTED WHICH RESULTS IN THE TERM BEING
8256*     OVERWRITTEN BY ANOTHER TERM.  IF THE TERM IS NON-0, W IS INC.
8257*          CALLED BY THE AND SUBROUTINE.
8258*************************************************************************
8260 C=1                     \*SET THE SPECIAL CHARACTER FLAG.
8270 D=1                     \*SET THE MINUS FLAG.
8280 Z$=""                   \*CLEAR A TEMPORARY VARIABLE.
8290 B$=""                   \*CLEAR ANOTHER TEMPORARY VARIABLE.
8300 INPUT :H;W,Y$           \*GET TERM TO BE ANALYZED.
8301 L=LEN(Y$)               \*COMPUTE  LENGTH OF TERM.
8310 FOR A=1 TO L            \*BEGIN TERM ANALYSIS.
8320     X$=Y$(:A,1)         \*EXAMINE A CHARACTER FROM Y$.
8330     IF X$>"Z" THEN 8350\*NUMERAL?
8332     IF X$="-" THEN 8350\*MINUS SIGN?
8334     IF X$="*" THEN 8350\*STUCK-AT-0 SYMBOL?
8336     IF X$="!" THEN 8350\*STUCK-AT-1 SYMBOL?
8340     IF C=1 THEN 8400\*END OF LITERAL?
8341     *YES.  LOOK FOR ITS COMPLEMENT.\GOTO 8420
8350     C=0                 \*SPECIAL SYMBOL FOUND.  CLEAR FLAG.
8360     IF X$<>"-" THEN 8370\*WAS THE SYMBOL A - ?
8365     D=0                 \*YES.  CLEAR FLAG.
8366     *GET NEXT CHARACTER.\GOTO 8600
8370     IF D<>1 THEN 8400\*CHECK FOR COMPLEMENTED LITERAL.
8390     Z$=Z$+"-"           \*NOT COMPLEMENTED.  INSERT MINUS.
8400     Z$=Z$+X$            \*CONSTRUCT TEST LITERAL.
8410     *GET NEXT CHARACTER.\GOTO 8600
8420     C=1                 \*SET SPECIAL CHARACTER FLAG.
8430     B1=A                \*SAVE POINTER TO Y$.
8440     FOR B=B1 TO L       \*FIND NEXT LITERAL IN TERM.
8450         INPUT :H;W,Y$   \*GET THE TERM FROM THE FILE.
```

```
8451          X$=Y$(:B,1)     \*GET A CHARACTER OF THE TERM.
8460          IF X$>"Z" THEN 8480\*A NUMERAL?
8462          IF X$="-" THEN 8480\*A MINUS SIGN?
8464          IF X$="*" THEN 8480\*STUCK-AT-0 SYMBOL?
8466          IF X$="!" THEN 8480\*STUCK-AT-1 SYMBOL?
8470          IF C=1 THEN 8490\*END OF LITERAL?
8471          *YES.   COMPARE TO TEST LITERAL.\GOTO 8510
8480          C=0             \*SPECIAL CHARACTER. CLEAR FLAG.
8490          B$=B$+X$        \*CONSTRUCT LITERAL.
8500          *GET NEXT CHARACTER.\*GOTO 8540
8510          IF Z$=B$ THEN 8620\*COMPARE WITH TEST LITERAL.
8520          C=1             \*NOT = .  SET SPECIAL CHAR FLAG.
8530          B$=""           \*CLEAR TEMPORARY VARIABLE.
8535          *SEARCH FOR NEXT LITERAL.\GOTO 8450
8540     NEXT B               \*REPEAT FOR NEXT CHARACTER.
8550     IF Z$=B$ THEN 8620   \*COMPARE LAST LITERAL WITH TEST.
8560     C=1                  \*SET SPECIAL CHARACTER FLAG.
8570     D=1                  \*SET THE MINUS SIGN FLAG.
8580     Z$=""                \*CLEAR A TEMPORARY VARIABLE.
8590     B$=""                \*CLEAR ANOTHER TEMP VARIABLE.
8595     *DEVELOP A NEW TEST LITERAL.\GOTO 8320
8600 NEXT A                   \*REPEAT FOR NEXT CHARACTER.
8610 W=W+1                    \*NON-0 TERM.  INC POINTER TO W.
8620 RETURN                   \*RETURN TO AND SUBROUTINE.
8650*****************************************************************
8651*         SUBROUTINE TO SORT THE LITERALS IN A PRODUCT TERM.
8652*    LITERALS ARE PASSED TO SUBROUTINE IN Z$ AND THE SORTED
8653*    TERM IS RETURNED IN Z$.
8654*         CALLED BY THE AND SUBROUTINE.
8655*         CALLS THE TERM DISASSEMBLY SUBROUTINE.
8656*****************************************************************
8660 *DISASSEMBLE THE TERM INTO LITERALS.\GOSUB 7800
8663 C=0                      \*INITIALIZE COUNTER.
8668 B1=A                     \*GET NUMBER OF LITERALS IN TERM FROM SUBR 7800
8670 FOR A=1 TO B1            \*SORT THE LITERALS IN A$ USING A BUBBLE SORT.
8680     IF A$(A+1)="." THEN 8740\*END OF LITERALS.
8690     IF A$(A)<A$(A+1) THEN 8730\*COMPARE TWO LITERALS.
8700     Z$=A$(A)             \*RE-ORDERING NEEDED.  SAVE ONE LITERAL.
8710     A$(A)=A$(A+1)        \*SWAP PLACES OF THE TWO.
8720     A$(A+1)=Z$           \*COMPLETE THE SWAP.
8730 NEXT A                   \*REPEAT FOR NEXT PAIR.
8740 IF B1=1 THEN 8750        \*CHECK FOR END OF SORT.
8745 B1=B1-1                  \*DECREASE LOOP LIMIT.  CONTINUE SORT.
8746 *REPEAT FOR NEXT PASS THROUGH THE LIST.\GOTO 8670
8750 FOR A=1 TO 20            \*SORT COMPLETE.  LOOK FOR DUPLICATE LITERALS.
8760     IF A$(A+1)="." THEN 8830\*LOOK FOR END OF LITERAL LIST.
8770     IF A$(A)=A$(A+1) THEN 8780\*LOOK FOR IDENTICAL LITERALS.
8771     *NOT IDENTICAL.  CONTINUE SEARCH.\GOTO 8820
8780     C=C+1                \*INC COUNTER.
8790     FOR B0=A TO 20       \*REMOVE DUPLICATE LITERAL FROM LIST.
8800          A$(B0)=A$(B0+1)\*SHIFT A LITERAL.
8805          IF A$(B0)="." THEN 8820\*END OF LIST?
```

```
8810   NEXT B0                \*CONTINUE SHIFTING.
8820 NEXT A                   \*REPEAT FOR NEXT PAIR.
8830 Z$=""                    \*CLEAR TEMPORARY VARIABLE.
8835 FOR A=1 TO 20            \*BEGIN RECONSTRUCTION OF PRODUCT TERM.
8840   IF A$(A)="," THEN 8870\*END OF LITERAL LIST?
8850   Z$=Z$+A$(A)            \*NO.  APPEND TO TERM.
8860 NEXT A                   \*REPEAT FOR NEXT LITERAL.
8870 RETURN                   \*RETURN AND SUBROUTINE.
9000*************************************************************
9001*       SUBROUTINE TO TRANSFER A FN FROM AN ARRAY TO A FILE.
9002*  FUNCTION IS ALWAYS TRANSFERRED FROM ARRAY H.  THE FILE IS
9003*  EITHER OLD1, OLD0, NEW0, OR NEW1 AS A FUNCTION OF J AND S.
9004*  CALLED BY THE MAIN PROGRAM (SIM2).
9005*************************************************************
9008 INPUT=0                  \*RETURN TO NORMAL INPUT MODE.
9010 CLOSE :G                 \*CLOSE FILE ASSIGNED TO CHANNEL G.
9012 ON G GOTO 9014,9016,9018,9020\*OPEN FILE TO CHANNEL G.
9014 OPEN "OLD1" TO :1, INPUT UPDATE
9015 *GOTO NEXT STEP          \GOTO 9022
9016 OPEN "OLD0" TO :2, INPUT UPDATE
9017 *GOTO NEXT STEP          \GOTO 9022
9018 OPEN "NEW0" TO :3, INPUT UPDATE
9019 *GOTO NEXT STEP          \GOTO 9022
9020 OPEN "NEW1" TO :4, INPUT UPDATE
9022 X$=N(I1,1)               \*GET FUNCTION NAME.
9025 IF J<>0 THEN 9040        \*CHECK FUNCTION FLAG.
9030 X$=X$+"-"                \*0-FN BEING TRANSFERED.  ADD - TO NAME.
9040 Z$=STR(T)                \*CONVERT TIME VARIABLE TO STRING.
9041 X$=X$+Z$(:2)             \*APPEND TIME TAG TO FN NAME.
9045 X$=X$+"="                \*APPEND = TO FUNCTION NAME.
9060 PRINT :J-2*S*J-S+3;Y5*(I1-1)+1,X$\*LOAD NAME IN FILE 1ST RCD.
9070 FOR X=1 TO 1000          \*BEGIN TRANSFER OF TERMS.
9075   INPUT :H;X,X$          \*GET TERM FROM ARRAY H.
9076   PRINT :J-2*S*J-S+3,X$\*PLACE TERM IN FILE.
9080   IF X$="." THEN 9100\*END OF FUNCTION?
9090 NEXT X                   \*NO.  REPEAT FOR NEXT TERM.
9100 INPUT=$                  \*CHANGE INPUT MODE.
9200 RETURN                   \*RETURN TO MAIN PROGRAM.
10000************************************************************
10001*       SUBROUTINE TO COMPARE NEW FUNCTION SET TO OLD SET.
10002*  FUNCTIONS ARE ACCESSED FROM THE OLD1, OLD0, NEW0, AND NEW1
10003*  FUNCTION FILES.  IF THE TWO SETS OF FUNCTIONS ARE EQUAL,
10004*  N0=1 IS RETURNED.  OTHERWISE, N0=0 IS RETURNED.
10005*       CALLED BY MAIN PROGRAM (SIM2).
10006************************************************************
10008 CLOSE :1\CLOSE :2\CLOSE :3\CLOSE :4\*CLOSE ALL OPEN FILES.
10010 OPEN "OLD1" TO :1, INPUT UPDATE
10011 OPEN "OLD0" TO :2, INPUT UPDATE
10012 OPEN "NEW0" TO :3, INPUT UPDATE
10013 OPEN "NEW1" TO :4, INPUT UPDATE
10015 FOR X=1 TO N1           \*BEGIN COMPARISON.
10020   INPUT :1+3*S;Y5*(X-1)+1,B$\*GET FN NAME.
```

```
10040   INPUT :4-3*S;Y5*(X-1)+1,C$\*GET FN NAME.
10045   INPUT :1+3*S,B$          \*GET TERM OF 1ST FN.
10047   INPUT :4-3*S,C$          \*GET TERM OF 2ND FN.
10050   IF B$=C$ THEN 10060 \*TERMS EQUAL?
10055   RETURN                   \*NO. THEN FNS NOT EQUAL.
10060   IF B$="." THEN 10080\*YES.  END OF FN?
10070   *CONTINUE COMPARISON.\GOTO 10045
10080   INPUT :2+S;Y5*(X-1)+1,J$\*GET FN NAME.
10100   INPUT :3-S;Y5*(X-1)+1,K$\*GET FN NAME.
10105   INPUT :2+S,J$            \*GET TERM OF 1ST FN.
10107   INPUT :3-S,K$            \*GET TERM OF 2ND FN.
10110   IF J$=K$ THEN 10120 \*TERMS EQUAL?
10115   RETURN                   \*NO.  THEN FNS NOT EQUAL.
10120   IF J$="." THEN 10140\*END OF FN?
10130   *NO.  CONTINUE COMPARISON.\GOTO 10105
10140 NEXT X                     \*REPEAT FOR NEXT FUNCTIONS.
10150 NO=1                       \*FUNCTION SETS IDENTICAL.
10160 RETURN                     \*RETURN TO MAIN PROGRAM (SIM2).
11000*****************************************************************
11001*       SUBROUTINE TO PRINT TEST-FUNCTION IDENTIFICATION AND
11002* TEST-FUNCTION PRODUCT TERMS IN LIST FORMAT.
11005*****************************************************************
11008 *PRINT HEADER INFO FOR UNSPECIFIED FAULTS MODE.
11009 INPUT=0
11010 PRINT"OUTPUT-LINE= ";O$
11011 PRINT"INPUT-LINE= ";W$
11012 PRINT"TIME-TAG =";T$
11020 PRINT
11030 PRINT"TEST-FUNCTION= ";
11049 *PRINT LIST OF TEST FUNCTION TERMS.
11050 FOR X=1 TO 200
11060       INPUT :H;X,X$
11065       IF X$="." THEN 11325
11070       PRINT TAB(20);X$
11080       PRINT
11090 NEXT X
11325 INPUT=$
11327 PRINT
11330 RETURN
17000*****************************************************************
17001*       SUBROUTINE TO MINIMIZE FUNCTIONS.
17002*  A FUNCTION IS SAID TO BE MINIMIZED (IN THIS CONTEXT) IF IT
17003*  CONTAINS NO REDUNDANT PRODUCT TERMS.  HENCE THE SUBROUTINE
17004*  REMOVES DUPLICATE TERMS AND TERMS COVERED BY ANOTHER TERM.
17005*       A FUNCTION IS PASSED TO THE SUBROUTINE IN ARRAY H.
17006*  THE REDUCED FUNCTION IS RETURNED IN ARRAY H.
17007*       CALLED BY THE AND SUBROUTINE AND THE OR SUBROUTINE.
17008*****************************************************************
17010 FOR X=1 TO X2-1            \*EXAMINE EACH TERM OF THE FN.
17015       INPUT :H;X,Z$  \*GET A TERM FROM ARRAY H.
17016       IF Z$="." THEN 17220\*ENDOF FUNCTION?
17020       X$=Z$                \*NO. COPY TERM TO X$ FOR PARSING.
```

```
17030          *PARSE TERM INTO LITERALS.\GOSUB 7800
17040          FOR Y=1 TO 20  \*COPY LITERALS TO ARRAY D$.
17050               D$(Y)=A$(Y)\*COPY A TERM.
17060               IF A$(Y)="." THEN 17080\*END OF LIST?
17070          NEXT Y         \*NO.  REPEAT FOR NEXT LITERAL.
17080          FOR Y=X+1 TO X2-1\*COMPARE ABOVE TERM TO ALL OTHER TERMS.
17090               INPUT :H;Y,Z$\*GET A TERM FROM ARRAY H.
17091               IF Z$="." THEN 17210\*END OF FN?
17100               X$=Z$     \*COPY TERM TO X$ FOR PARSING.
17110               *PARSE THE TERM INTO LITERALS\GOSUB 7800
17115               F5=1           \*INITIALIZE FLAG INDICATING
17116                              *A REDUNDANT TERM.
17120               FOR B=1 TO 20\*COMPARE LITERAL LISTS OF TWO TERMS.
17122                    IF F5=0 THEN 17200\*CHECK FLAG FOR NO REDUNDANCY.
17125                    IF D$(B)="." THEN 17170\*END OF LIST?
17130                    FOR A=1 TO 20\*SCAN LIST OF 2ND TERM.
17135                         IF A$(A)="." THEN 17165\*END OF LIST?
17140                         IF A$(A)=D$(B) THEN 17160
17141                              *COMPARE LITERALS OF TWO LISTS.
17150                         F5=0 \*LITERALS ARE NOT ALIKE.
17151                              *CLEAR FLAG TO INDICATE NO REDUNDANCY.
17155                    NEXT A         \*REPEAT FOR NEXT LITERAL OF 2ND TERM
17160                    F5=1           \*MATCH FOUND. POSSIBLE REDUNDANCY.
17165               NEXT B              \*REPEAT FOR NEXT LITERAL OF 1ST TERM
17170               FOR C=Y TO X2-1\*2ND TERM REDUNDANT.  DELETE FROM H.
17180                    INPUT :H;C+1,Z$\*GET A TERM FROM H.
17185                    PRINT :H;C,Z$  \*PLACE TERM IN NEW POSITION.
17190               NEXT C              \*REPEAT FOR NEXT TERM.
17195               *GET NEW 2ND TERM.\GOTO 17090
17200          NEXT Y                   \*REPEAT FOR NEXT TERM.
17210     NEXT X                        \*REPEAT FOR NEXT FIRST TERM.
17220     RETURN                        \*RETURN TO CALLING PROGRAM.
20000     END
```

☆U.S. GOVERNMENT PRINTING OFFICE 1979—640-247/278 REGION NO' 4